

SoTA: Grupo 5a

Metaclásificadores

Random Forest, Rotation Forest, Bagging, Boosting, Cascading e Híbrido

Silvia Arenales Muñoz, Manuel Castillo Sancho, Rubén González Velasco,
Sergio Martos Pariente y Héctor Sanz González

9 de Enero de 2025

MUIA

índice

Introducción

Bagging

Boosting

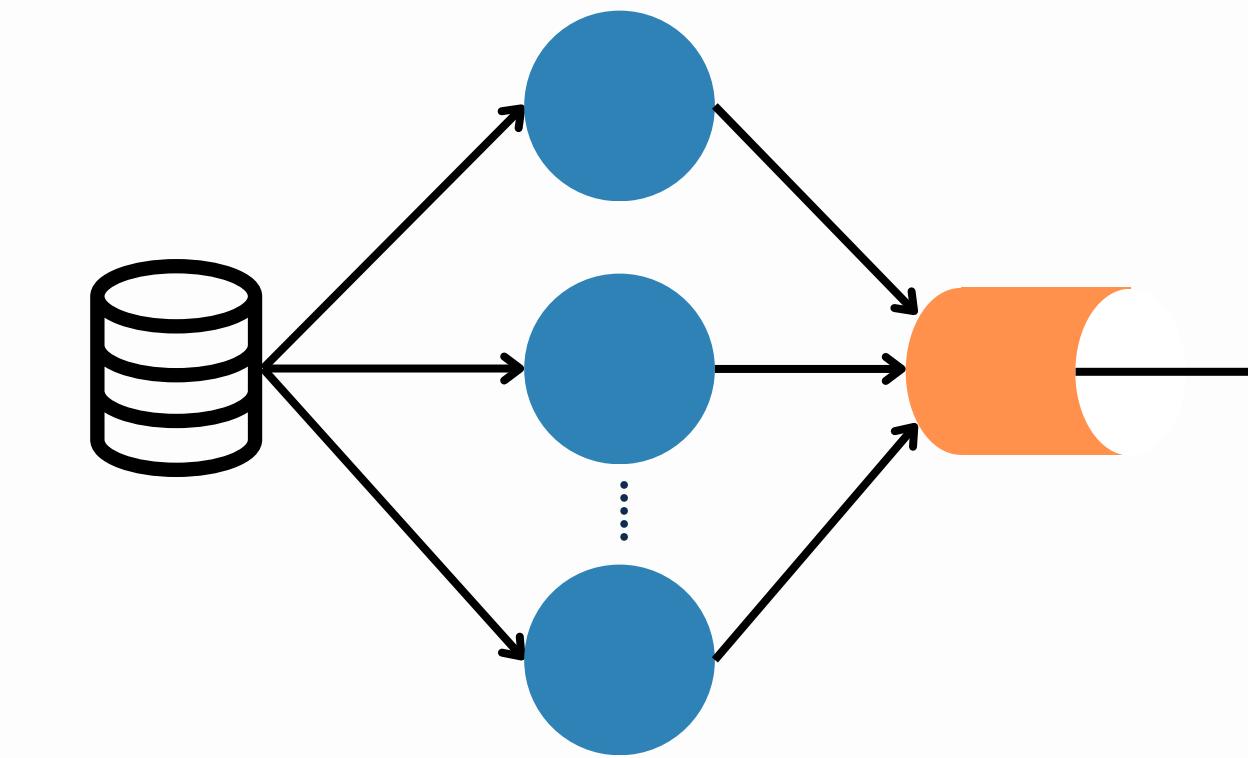
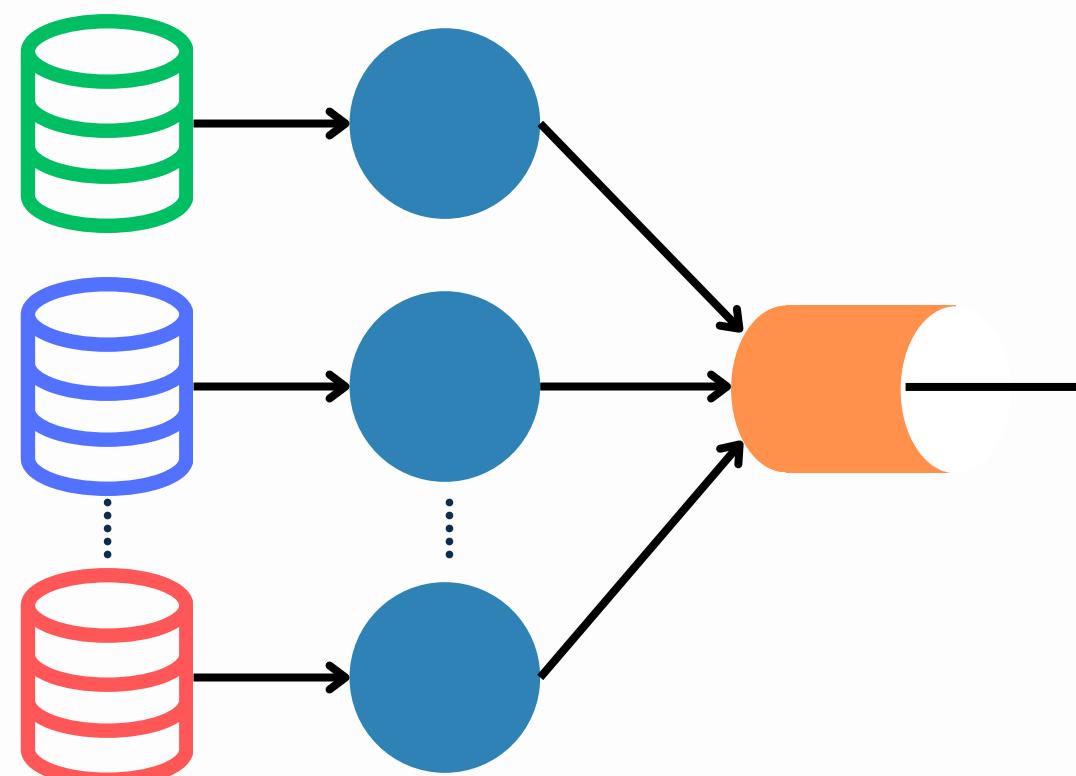
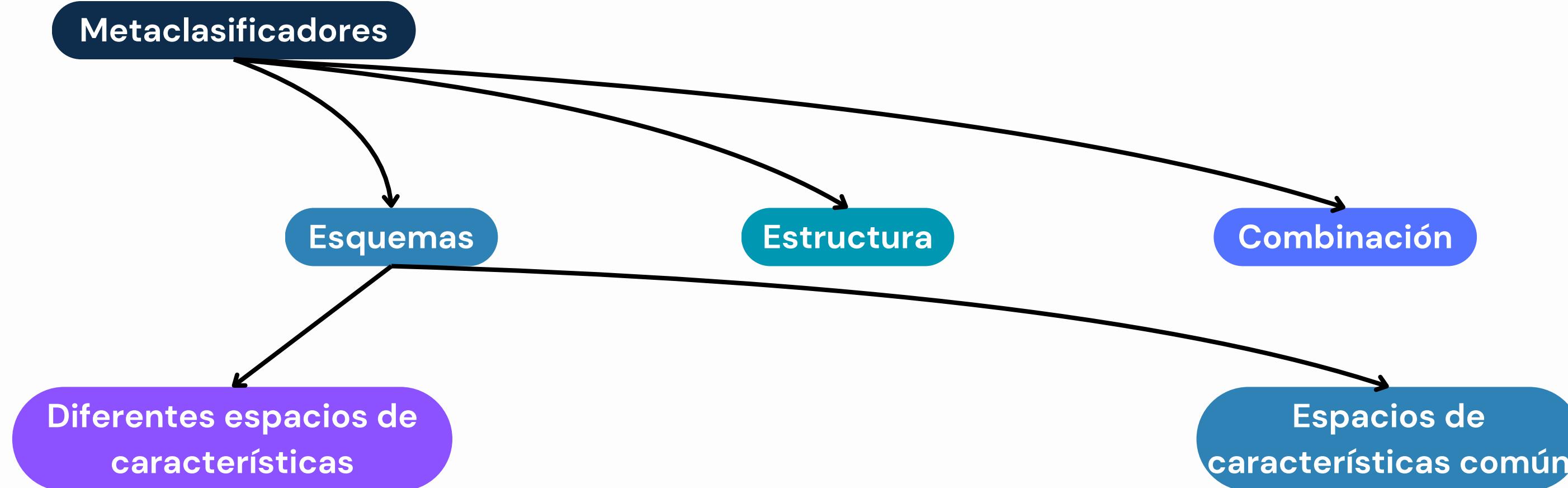
Random Forest

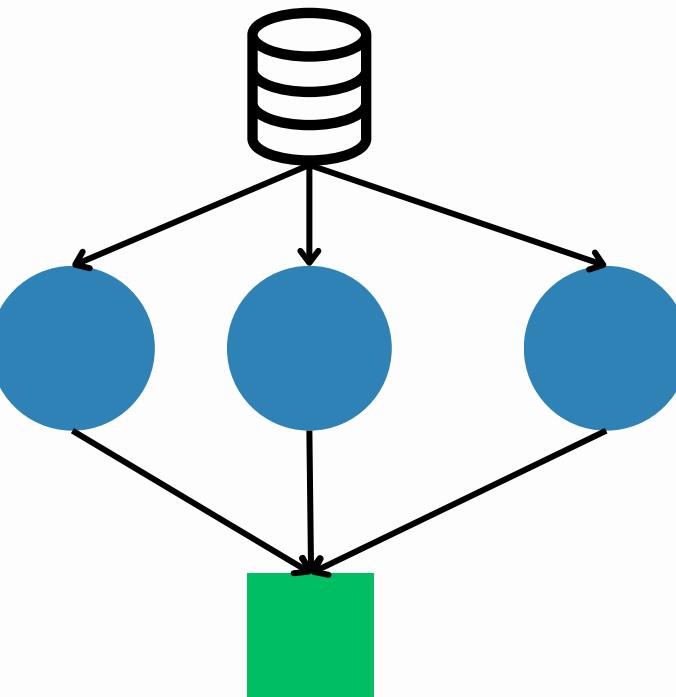
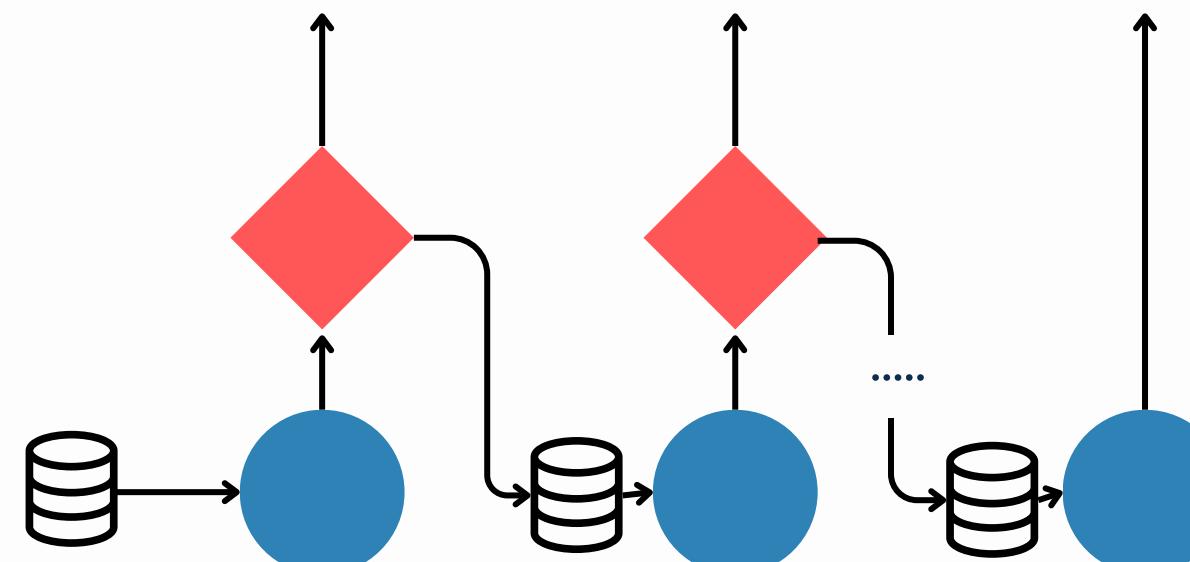
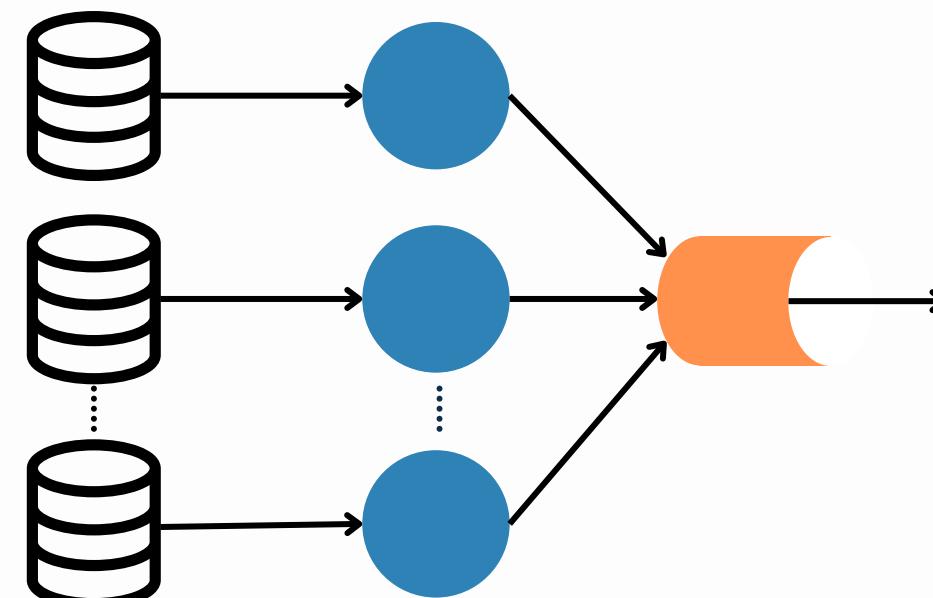
Rotation Forest

Cascading

Híbridos

Conclusiones y líneas futuras



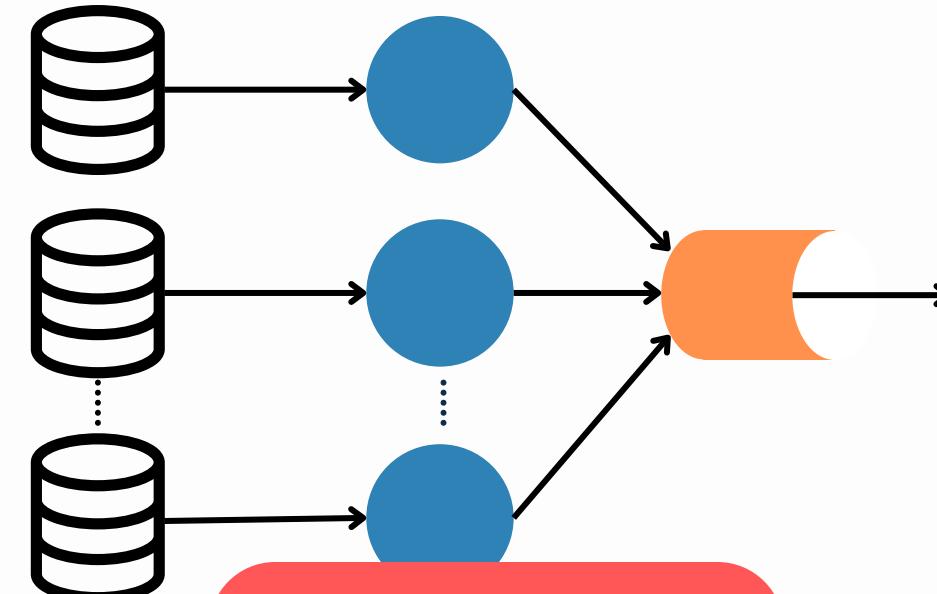
Metaclasificadores**Esquemas****Estructura**
Espacios de
características común**Combinación****Paralela****Secuencial****Jerárquica**

Metaclasificadores

Esquemas

Espacios de
características común

Paralela

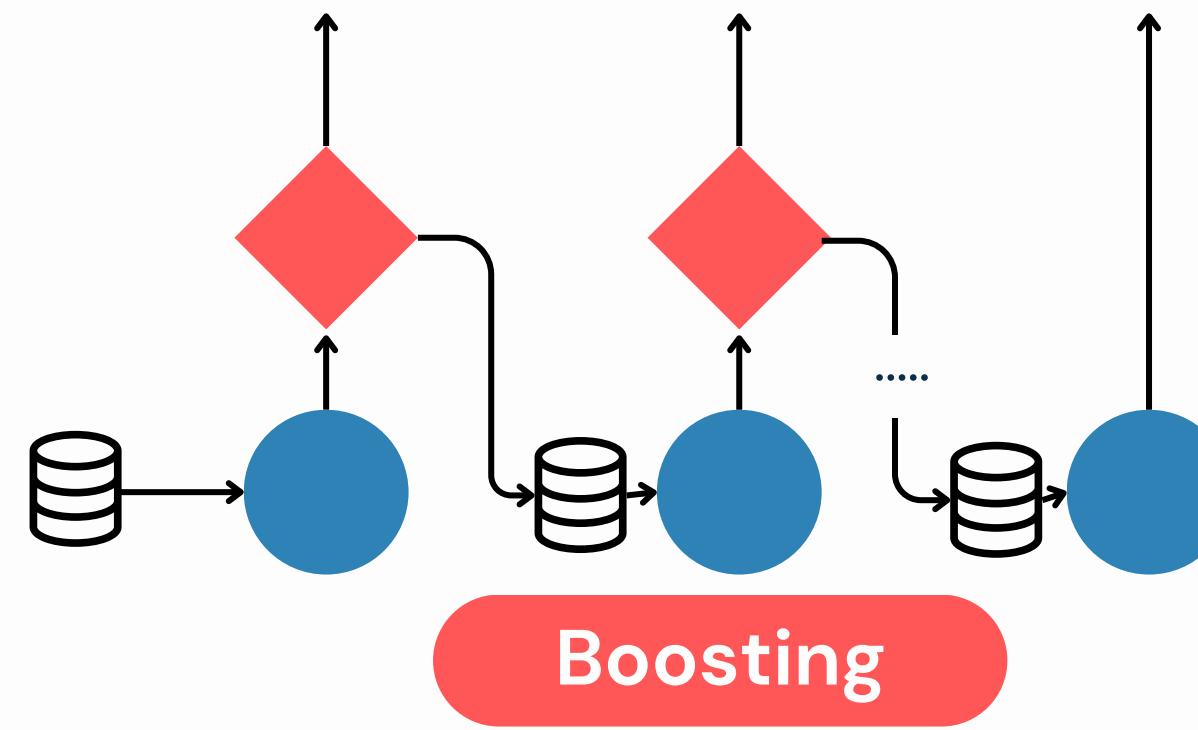


Bagging

Random Forest

Estructura

Secuencial

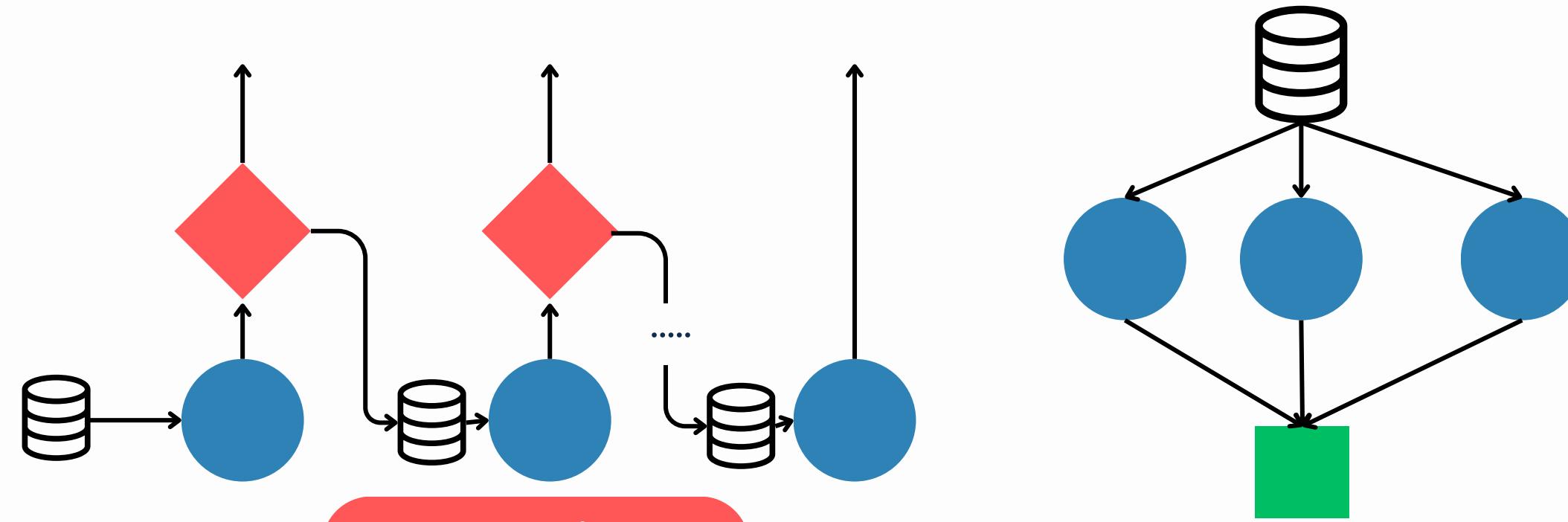


Boosting

Rotation Forest

Combinación

Jerárquica



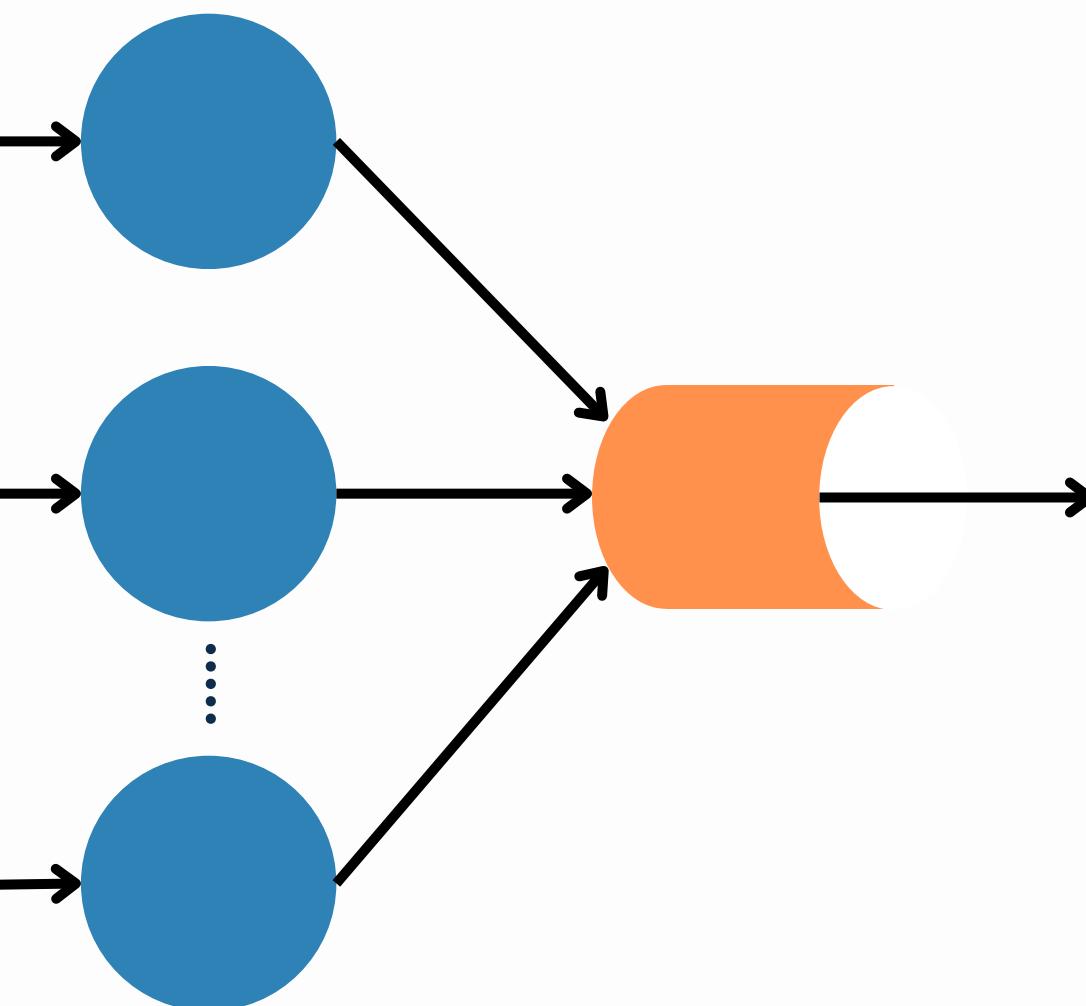
Metaclásificadores

Esquemas

Espacios de características comú

Estructura

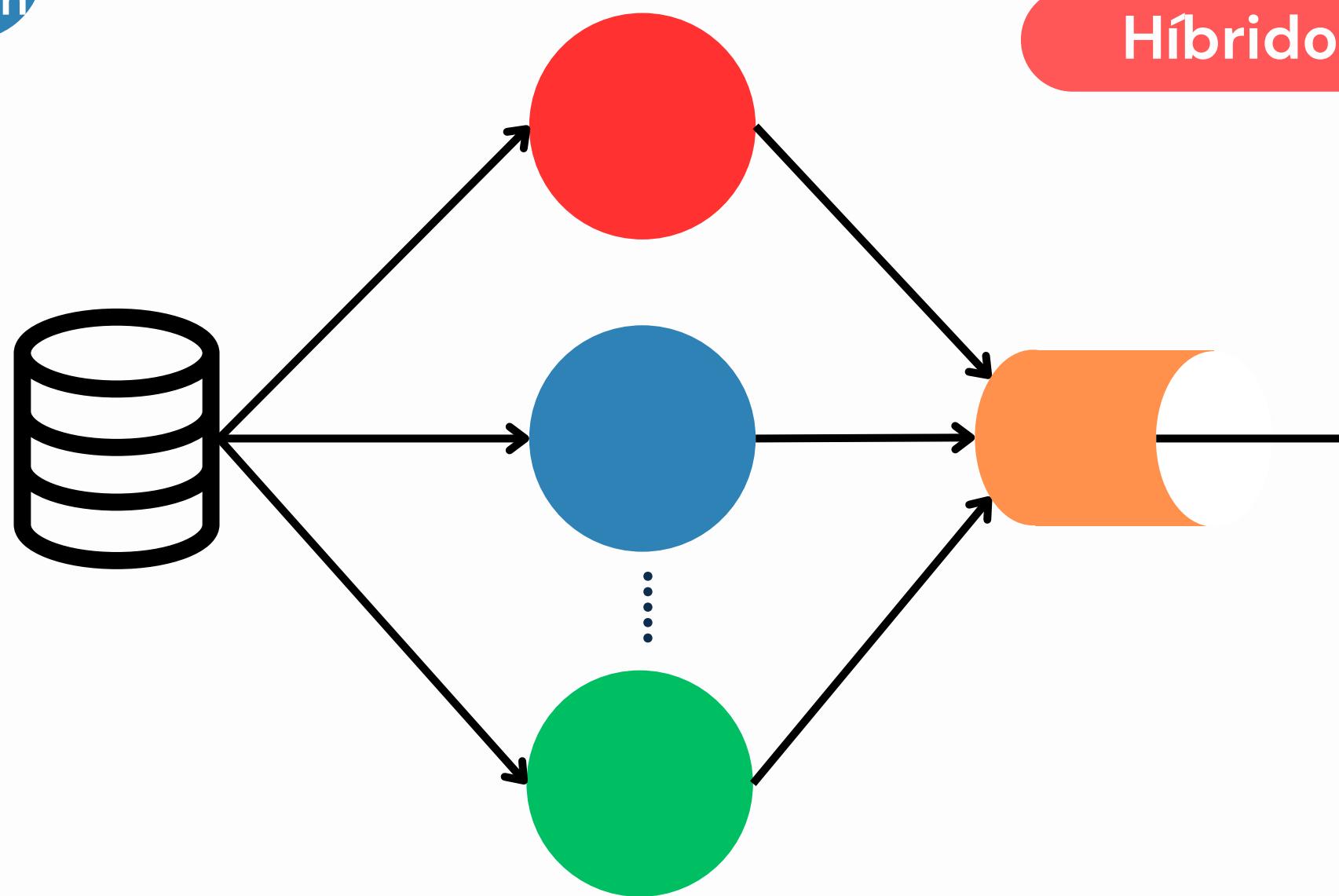
Combinación



- Fusión de etiquetas del output.
- Fusión de valores continuos del output.
- Stacking
- Cascading

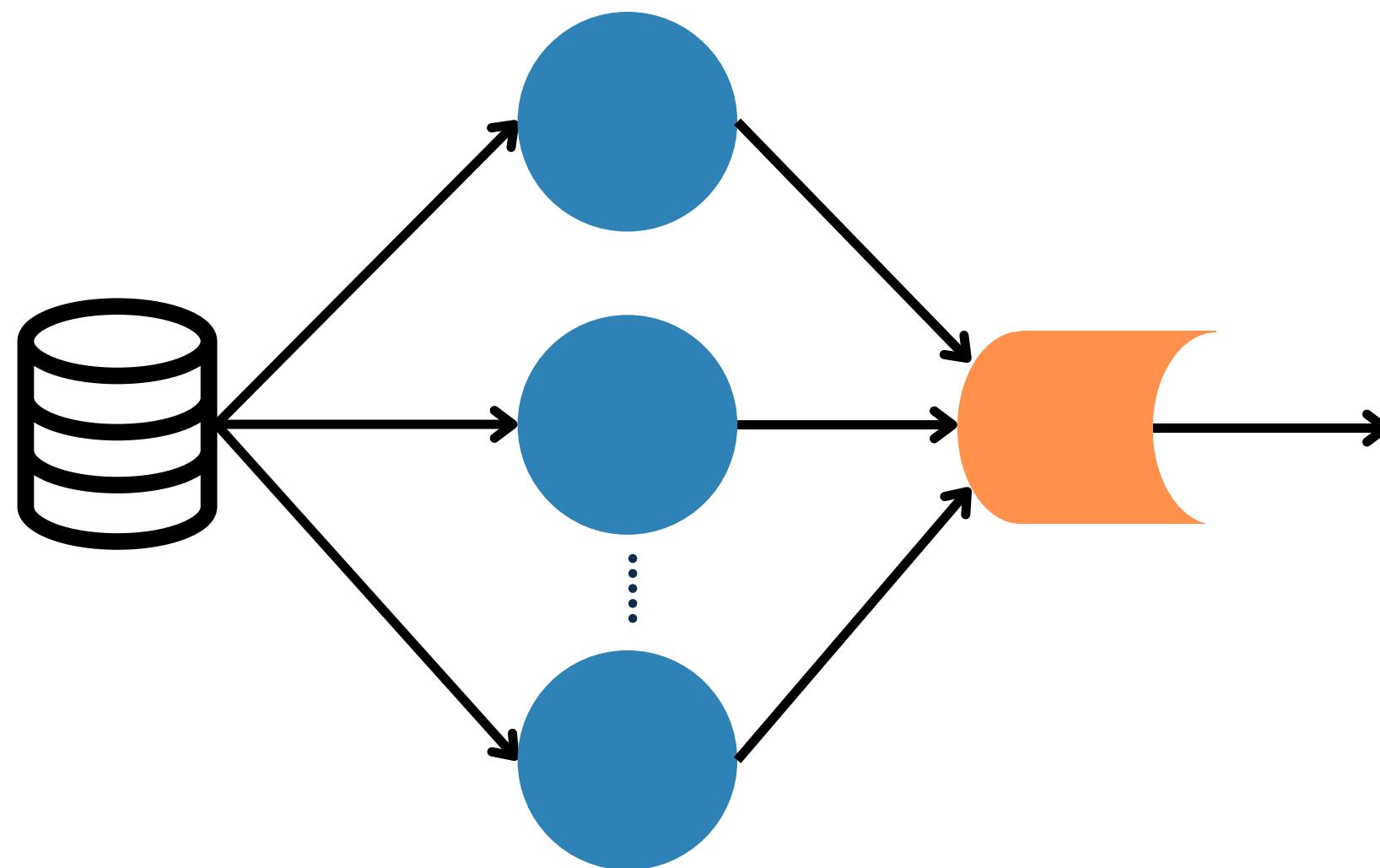
- Voto por mayoría
- Mayoría simple
- Voto por mayoría con umbral
- Voto por mayoría con pesos

- Media aritmética
- Mínimo
- Máximo
- Mediana
-

Metaclásificadores**Esquemas****Estructura****Combinación****Espacios de
características común****Híbridos**

Bagging

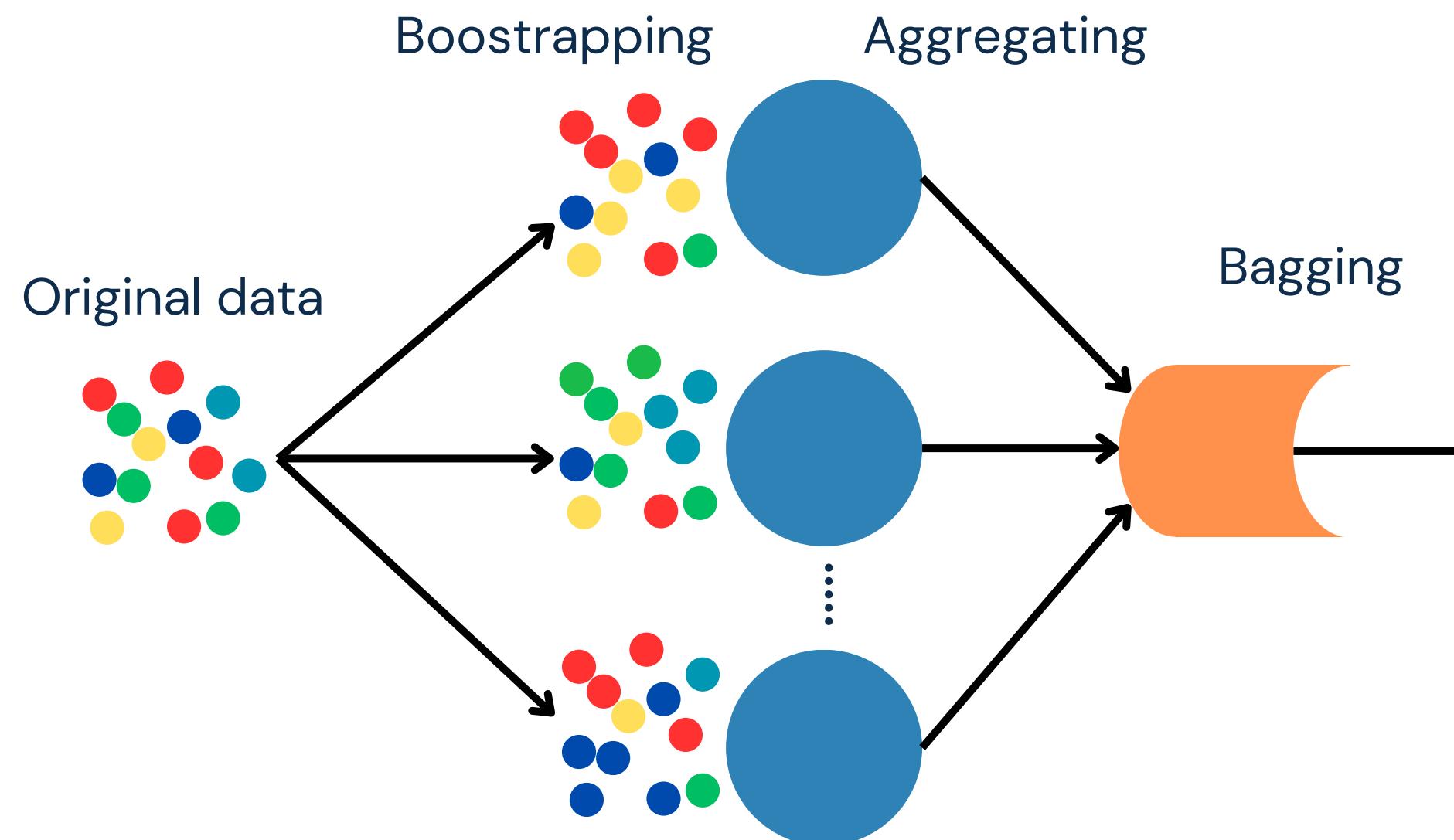
Breiman 1996



Crea múltiples versiones del modelo base entrenamiento en diferentes subconjuntos del conjunto de datos original.

Bagging

Breiman 1996



FASES:

1. Bootstrap Sampling

2. Aggregating

3. Bagging

Algorithm 1: Bagging

Input: Datos de entrenamiento \mathcal{D} , número de modelos base B , modelo base ϕ
Output: Conjunto de modelos $\{\phi_b\}_{b=1}^B$ y predicción combinada

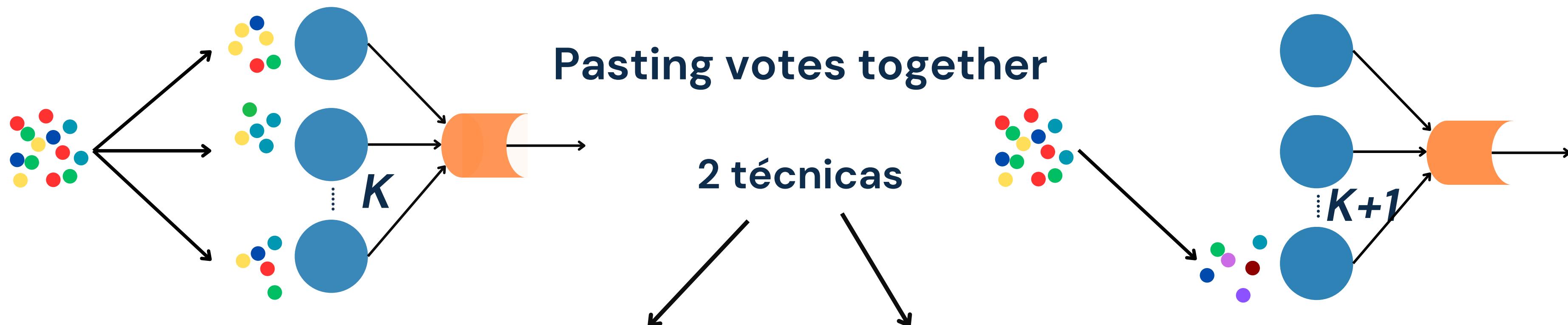
- 1 **for** $b = 1, \dots, B$ **do**
 - 2 - Generar una muestra bootstrap \mathcal{D}_b a partir de \mathcal{D} ;
 - 3 - Entrenar el modelo base ϕ_b con \mathcal{D}_b ;
- 4 **Predicción:** Para una nueva instancia x , combinar las predicciones $\{\phi_b(x)\}$ mediante:
 - **Clasificación:** Votación mayoritaria: $\hat{y} = \arg \max_y \sum_{b=1}^B \mathbb{I}(\phi_b(x) = y)$.
 - **Probabilidades Posteriore:**
 - Opcion 1: Usar las etiquetas predichas (con mayor probabilidad) y votar posteriormente.
 - Opcion 2: Promediar las probabilidades obteniendo nuevas probabilidades para cada clase y elegir aquella con la mayor probabilidad: $\hat{y} = \arg \max_y \frac{1}{B} \sum_{b=1}^B P(y|\phi_b(x))$.
 - **Regresión:** Promedio simple: $\hat{y} = \frac{1}{B} \sum_{b=1}^B \phi_b(x)$.

Variantes de Bagging

Pasting

Breiman 1999

Manejo de bases de datos extremadamente grandes, donde no es posible mantener todo el conjunto de datos en memoria, para el entrenamiento de los clasificadores se toman muestras de tamaño modesto M



- **Pasting Rvotes:** El conjunto de datos se obtiene por muestreo aleatorio sin reemplazo, el conjunto se conoce como Rprecinct
- **Pasting Ivotes:** El conjunto de datos se obtiene por por importancia, selecciona más frecuentemente aquellas instancias que tienen mayor probabilidad de ser clasificadas incorrectamente, el conjunto se conoce como Rprecinct

Variantes de Bagging

Adaptative

Breiman 1999

Bagging

Busca reducir tanto la varianza como el sesgo del modelo en problemas de regresión a través de un procedimiento iterativo y adaptativo.

1. Etapas iterativas:

- Se realiza bagging con un número fijo K de predictores.
- Mismos valores de entrada pero alterando los valores de salida en cada iteración.
- los valores de la variable de salida utilizados en la etapa j de bagging como $y_n^{(j)}$, siendo $y_n^{(1)}$ los valores de salida iniciales.

2. Actualización de los valores de salida:

- En la etapa j , denotemos por $\hat{y}_{n,k}$ los valores predichos dados por el k -ésimo predictor.
- Los valores de salida se actualizan como:

$$y_n^{(j+1)} = y_n^{(j)} - \text{average}_k \hat{y}_{n,k}, \quad (1)$$

donde el promedio sobre $\hat{y}_{n,k}$ se toma solo sobre aquellos k tales que la n -ésima instancia no está incluida en el conjunto de entrenamiento k .

3. Predicción para nuevas entradas:

Si x es un valor de entrada que no está en el conjunto de entrenamiento inicial, entonces el predictor $\phi^{(j+1)}(x)$ después de j etapas se obtiene como:

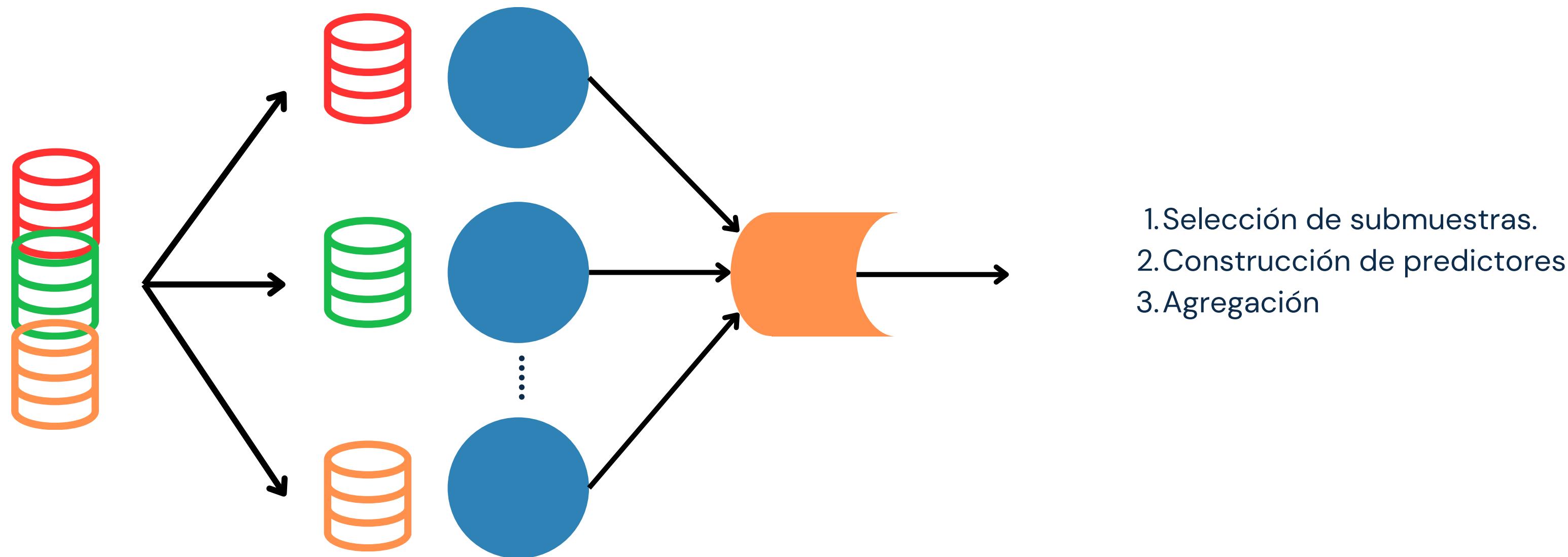
$$\phi^{(j+1)}(x) = \phi^{(j)}(x) + \text{average}_k \phi_{j,k}(x), \quad (2)$$

donde $\phi_{j,k}(x)$ es el valor predicho para x por el k -ésimo predictor en la etapa j .

Variantes de Bagging Subagging

Bühlmann and Yu
2002

Utiliza submuestreos sin reemplazo en lugar de muestras bootstrap completas para construir predictores agregados..



Presentador 2



POLITÉCNICA

UNIVERSIDAD
POLÍTÉCNICA
DE MADRID



ESCUELA TÉCNICA
SUPERIOR DE INGENIEROS
INFORMÁTICOS

Variantes de Bagging

Bragging

Bühlmann and Yu
2003

En lugar de calcular la media de las predicciones, como se realiza en el bagging tradicional, el bragging emplea la **mediana** como agregador

$$\phi_{n;Brag} = \text{median}(\{\phi_b(x); b = 1, \dots, B\})$$

Al usar la mediana, el bragging reduce el impacto de valores extremos en las predicciones, proporcionando un modelo final más robusto frente a ruido y valores atípicos en los datos.

Además de la mediana, se han explorado otros estimadores robustos, como el estimador de Huber y el estimador M con recorte de Hampel. Sin embargo, los estudios han demostrado que **la mediana ofrece un mejor rendimiento en términos de robustez y estabilidad.**

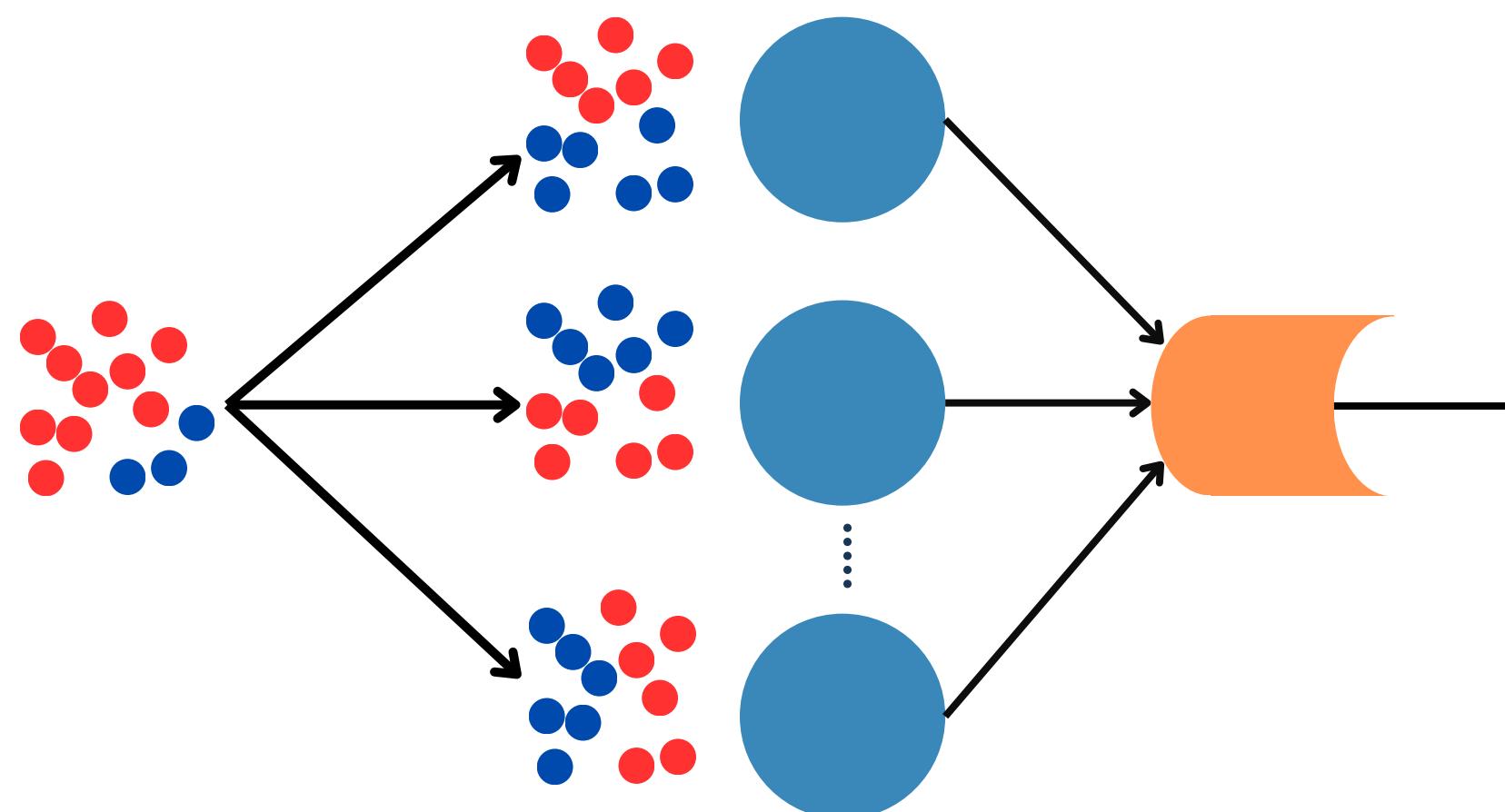
Variantes de Bagging

Under-Bagging

Wallace et al.
2011

Diseñada específicamente para **abordar el problema del desequilibrio de clases** en conjuntos de datos en un problema de clasificación binaria.

Este problema surge cuando una clase están representada significativamente menos que la otra, lo que puede **sesgar** los modelos predictivos **hacia la clase mayoritarias**.



El under-bagging combina los **principios de bagging** con **under-sampling**, un muestreo selectivo que submuestra las clases mayoritarias.

Objetivo: Equilibrar representación de las clases en cada muestra bootstrap.

Variantes de Bagging

Online Bagging

Oza and Russell
2001

Diseñada para escenarios en los que los datos llegan de manera **continua** o en **flujos** (data streams).

Permite entrenar modelos de **manera incremental** sin la necesidad de almacenar y procesar múltiples veces el conjunto de datos completo, lo que lo hace especialmente útil para grandes volúmenes de datos.

Bagging Tradicional → Las veces que una instancia aparece en una muestra bootstrap sigue una distribución binomial.

Si $N \rightarrow \infty$ entonces la distribución tiende a $\text{Po}(1)$

Algorithm 2: Online Bagging

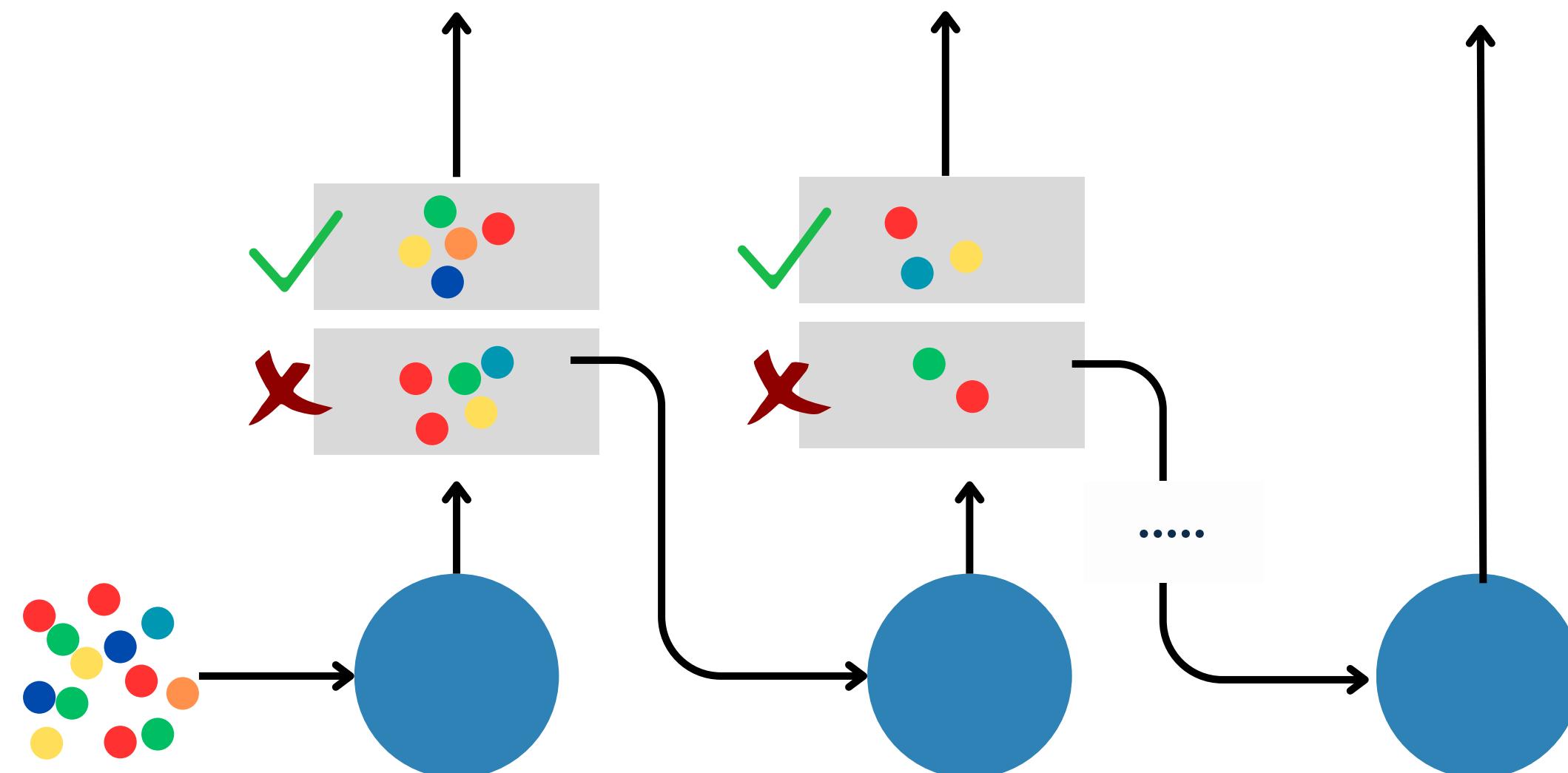
Input: Conjunto de modelos base $\{\phi_b\}_{b=1}^B$, nueva instancia d

- 1 **for** $b = 1, \dots, B$ **do**
- 2 - Generar $k \sim \text{Poisson}(1)$;
- 3 - Actualizar ϕ_b con d un total de k veces;

Output: Modelos base actualizados $\{\phi_b\}_{b=1}^B$

Boosting

Freund and Shapire 1997



IDEA:

Centrarse en los ejemplos “difíciles”,
que son mal clasificados por los
modelos anteriores

Arquitectura **SECUENCIAL**

Clasificadores débiles (error < 0.5)

Página 17 de 57

Boosting: AdaBoost

Freund and Shapire 1997

Pesos:

$$W_0 = 1/N \text{ (uniforme)}$$

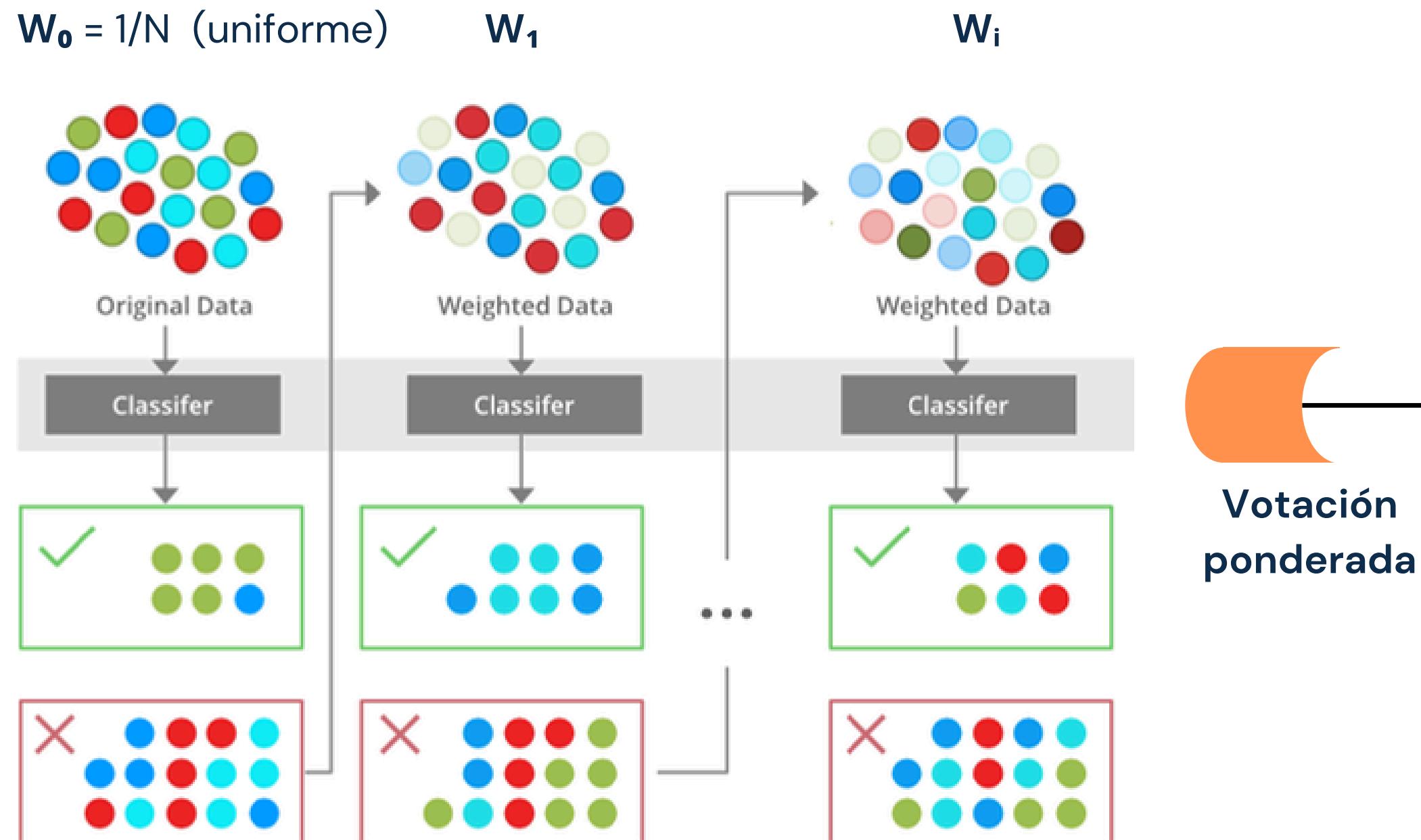


Imagen: geeksforgeeks

Muestreo adaptativo

El clasificador añadido en el paso i se entrena con una **muestra** del dataset. La distribución inicial del muestreo es uniforme (como en bagging).

Cada paso i actualiza esta distribución, aumentando la probabilidad de las instancias mal clasificadas en el paso $i - 1$, a fin de forzar al algoritmo base (o débil) a centrarse en los ejemplos más difíciles en sucesivas ejecuciones.

Al final, los modelos débiles se combinan en una **votación ponderada** : $\sum W_i \cdot \phi_i$

¿Por qué funciona AdaBoost?

El éxito de AdaBoost radica en su capacidad para reducir rápidamente el error de entrenamiento del ensamble a cero.

El error de entrenamiento decrece exponencialmente al agregar clasificadores

Freund and Shapire 1997

El error de prueba sigue disminuyendo incluso cuando el error de entrenamiento ya es cero.

Bartlett et al. 1998

AdaBoost es difícil de integrar en modelos estadísticos tradicionales → **LogitBoost**

Collins et al. 2002

AdaBoost es muy sensible al ruido, lo que puede causar sobreajuste → **BrownBoost**

Freund 2001

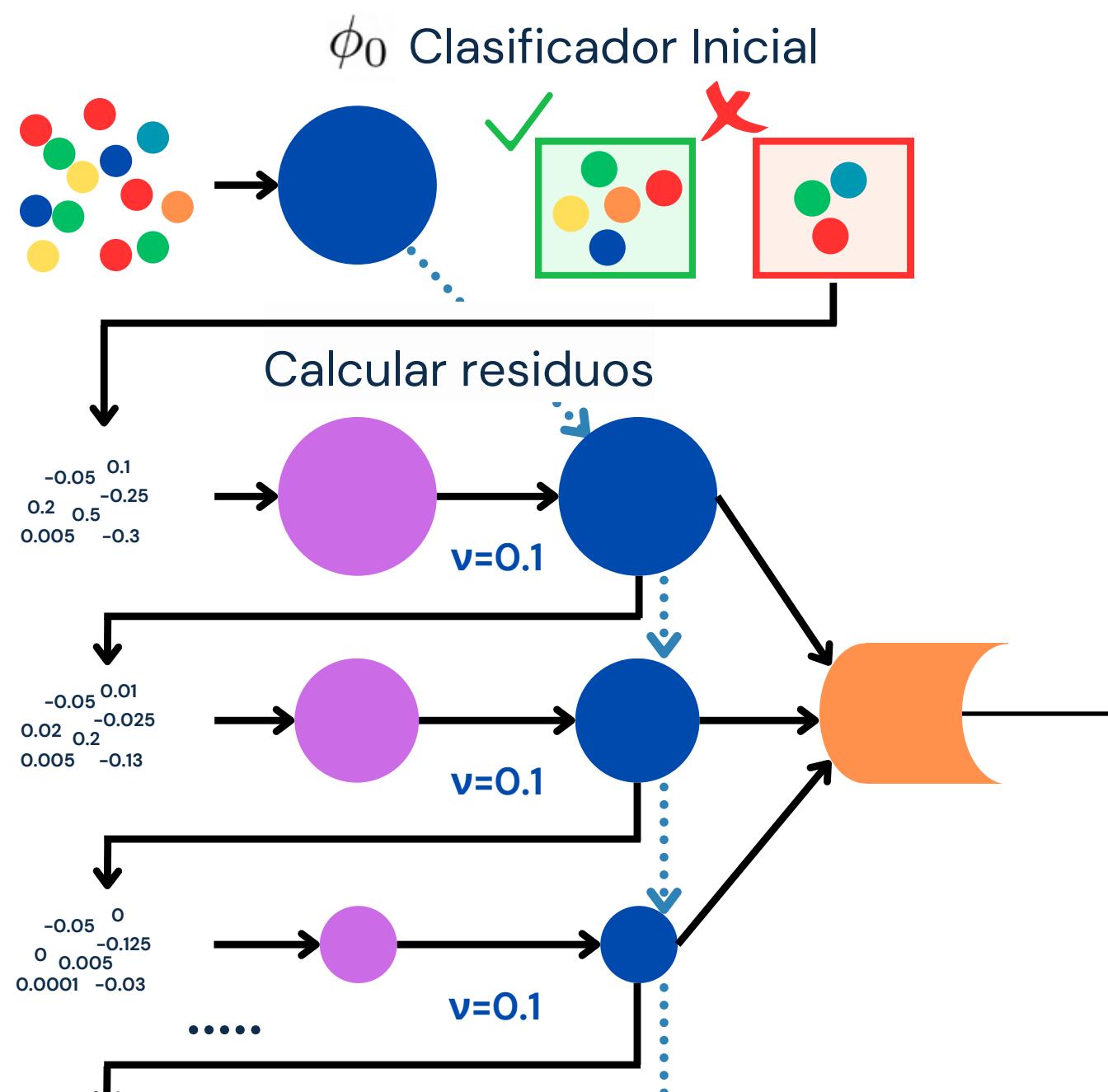
Variantes de Boosting: Gradient Boosting

Friedman 2001

Input:

- Función de Pérdida $L(y_i, F(x))$, *log-loss para clasificación*
- n° Iteraciones T
- Learning rate ν

(Breiman, 1997): El boosting puede interpretarse como un algoritmo de optimización sobre una función de coste adecuada.



En lugar de cambiar los pesos de los puntos de datos como AdaBoost, el boosting de gradiente se basa en los **pseudo-residuos** del predictor anterior.

Los aprendices débiles **h** (normalmente árboles), se construyen de manera greedy, y se entrenan (por regresión) para predecir los pseudo-residuos de los clasificadores ϕ_i , que por diseño, son el **gradiente negativo de la función de pérdida** respecto a las predicciones.

El modelo final es:

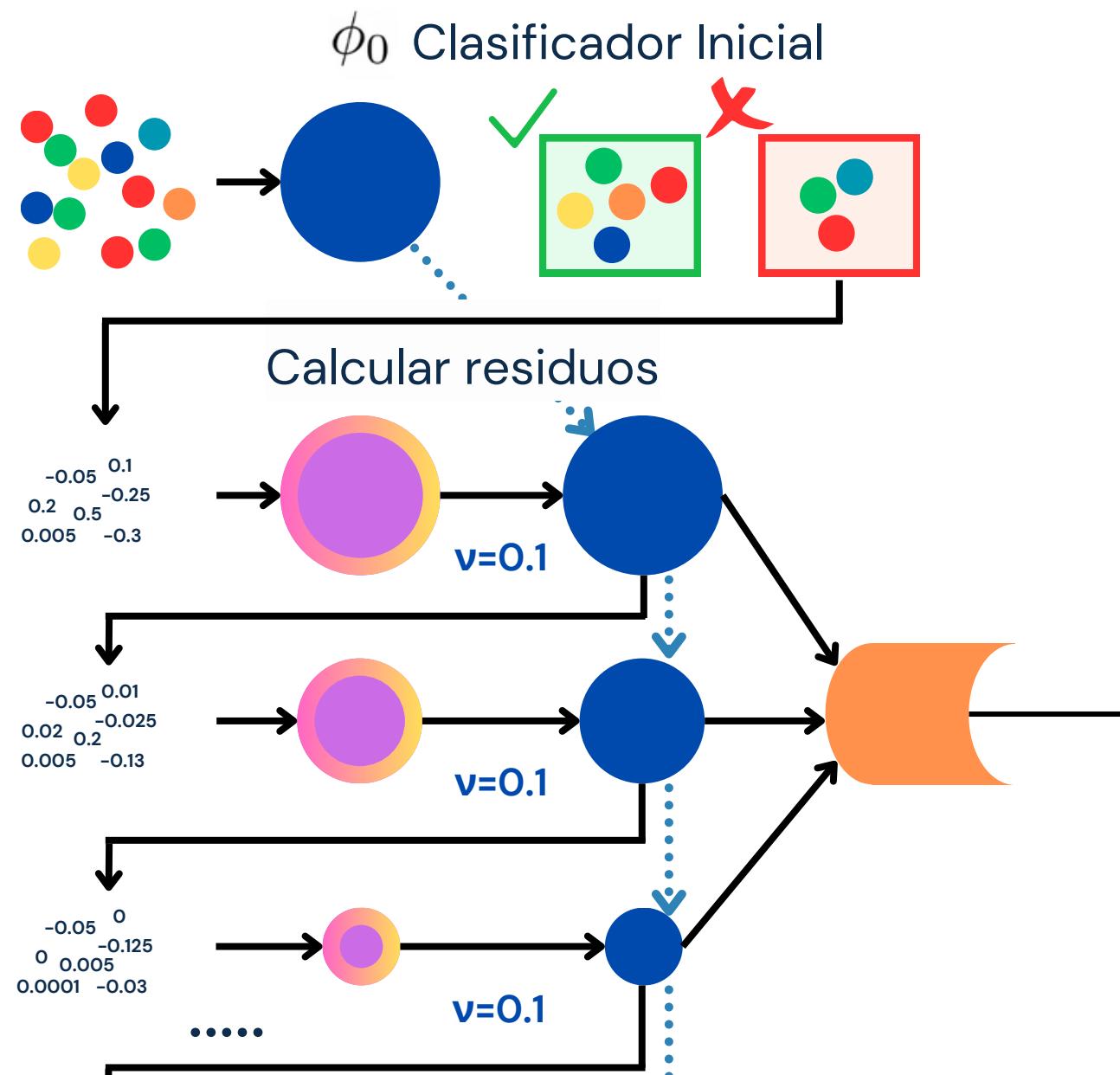
$$\phi_T(x) = \phi_0(x) + \sum_{t=1}^T \nu \cdot \underbrace{\gamma_t h_t(x)}_{\phi_t(x)}$$

Variantes de Boosting: Stochastic GB

Friedman 2002

Input:

- Función de Pérdida $L(y_i, F(x))$, *log-loss para clasificación*
- n° Iteraciones T
- Learning rate v
- Tamaño de muestra



Stochastic Gradient Boosting (SGB or Gradient Boosting Machines)

SGB funciona igual que Gradient Boosting, salvo que en SGB cada árbol se entrena utilizando un **subconjunto aleatorio de datos**, lo que introduce un elemento de aleatoriedad que ayuda a reducir la varianza y mejorar la generalización.

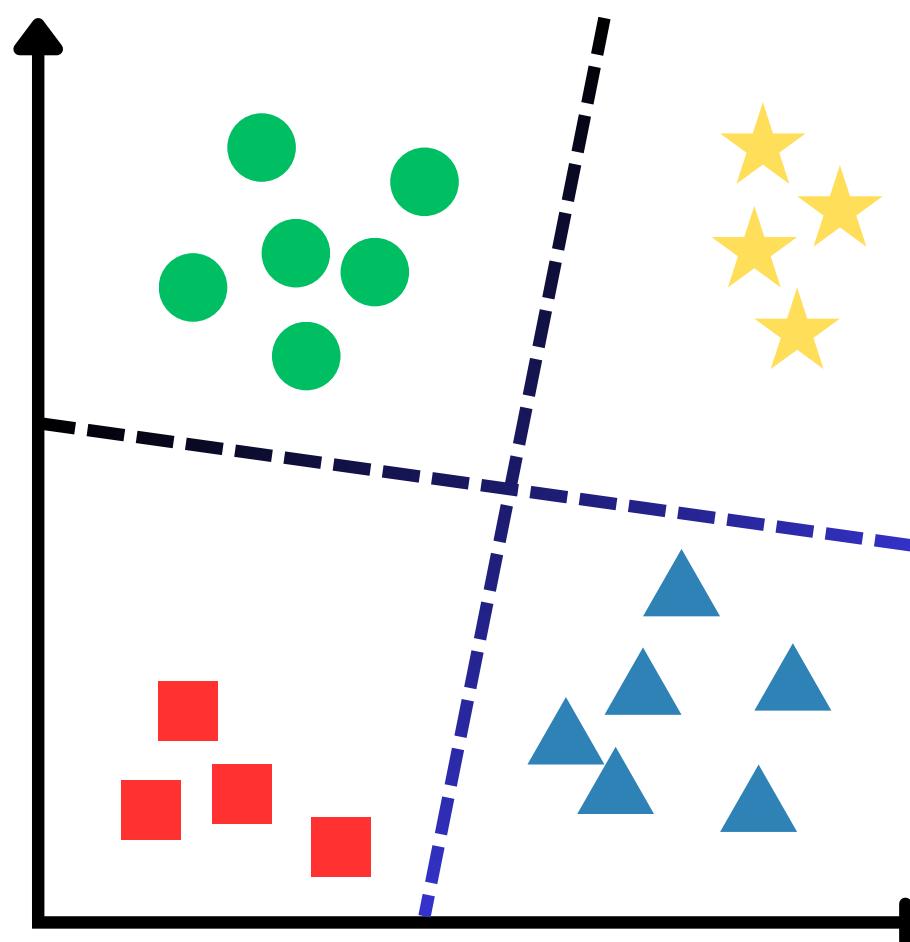
Implementaciones eficientes de GB y SGB:

- **XGBoost:** Extreme Gradient Boosting ([Chen and Guestrin, 2016](#))
- **LightGBM:** Light Gradient Boosting Machine ([Ke et al., 2017](#))
- **CatBoost:** Categorical Boosting ([Prokhorenkova et al., 2018](#))

Variantes de Boosting: Multiclase

Métodos Clásicos de Boosting Multiclase:

- AdaBoost.M1 Usa clasificadores débiles multiclas. M1 requiere $\varepsilon_i \geq 0.5$ [\(Freund and Shapire, 1997\)](#)
- SAMME Elimina la necesidad de superar el azar en todas las clases. [\(Hastie et al., 2009\)](#)
- AdaBoost.MM Utiliza una condición mínima de aprendizaje débil. [\(Mukherjee and Schapire, 2010\)](#)



Gradient Boosting para Problemas Multiclas

Se modifica la función de pérdida, utiliza la función de pérdida logística multinomial. Implementaciones como **XGBoost**, **LightGBM** y **CatBoost** tienen soporte multiclas nativo.

EN LA PRÁCTICA:

- XGBoost
 - LightGBM
 - CatBoost
- Más utilizados

- SAMME

Casos específicos

Presentador 3



POLITÉCNICA

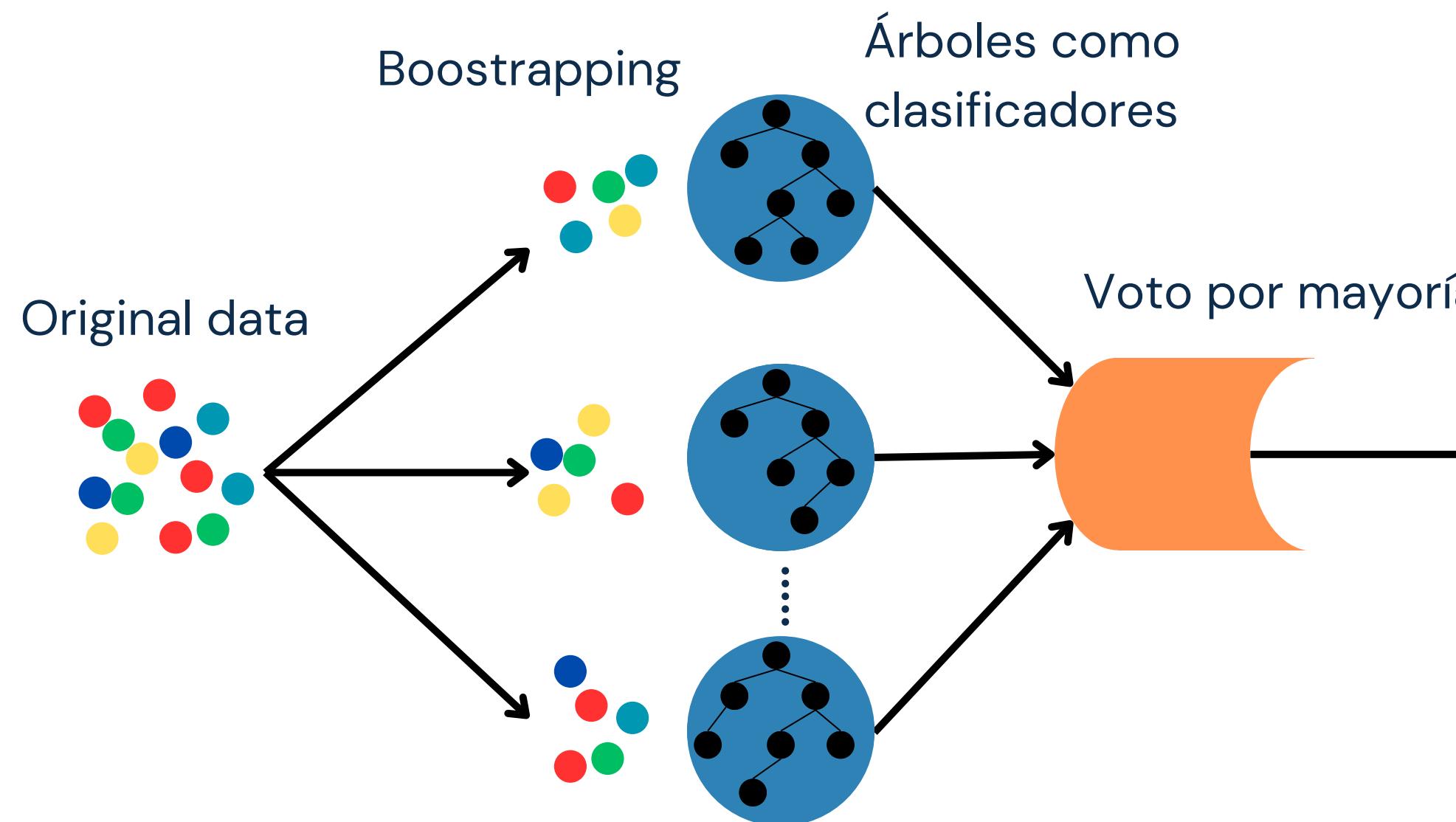
UNIVERSIDAD
POLÍTÉCNICA
DE MADRID



ESCUELA TÉCNICA
SUPERIOR DE INGENIEROS
INFORMÁTICOS

Random Forest

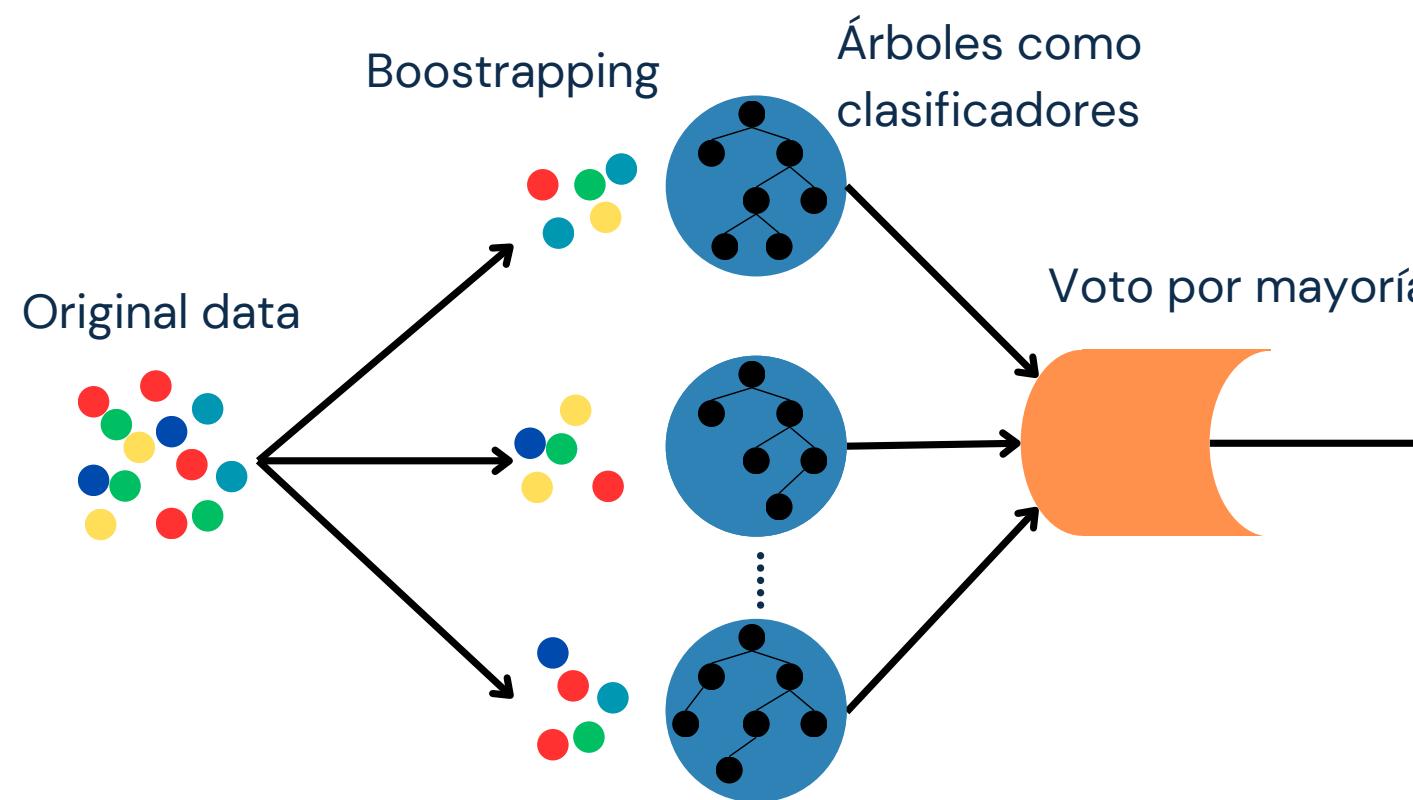
Breiman 2001



- Construir una colección de árboles con varianza controlada.
- Basado en Bagging.
- Mejora precisión y robustez.
- Implementación de la aleatoriedad adicional en el proceso de construcción de árboles.
- Aplicable tanto a estables como no estables, gracias al aleatorizar.

Random Forest

Breiman 2001



1. Muestreo Bootstrap.

2. Aleatoriedad en las variables y en las instancias.

3. Voto por mayoría.

Algorithm 2: Random Forest para Clasificación

Input: N(número de casos prueba), M(número de variables), B(número de árboles)

Output: Conjunto de árboles de decisión $\{T_b\}_1^B$

1 $m = \sqrt{M}$

2 for $b=1, \dots, B$ do

3 (i) Extraer una muestra de bootstrap de tamaño N (con reemplazo) del conjunto de entrenamiento. Usar las instancias no seleccionadas (*out-of-bag*) para evaluar el error.

4 (ii) Crecer un árbol de decisión binario T_b para los datos de bootstrap de la siguiente manera, hasta que se cumpla el criterio de detención:

5 - Seleccionar aleatoriamente m variables del conjunto de características.

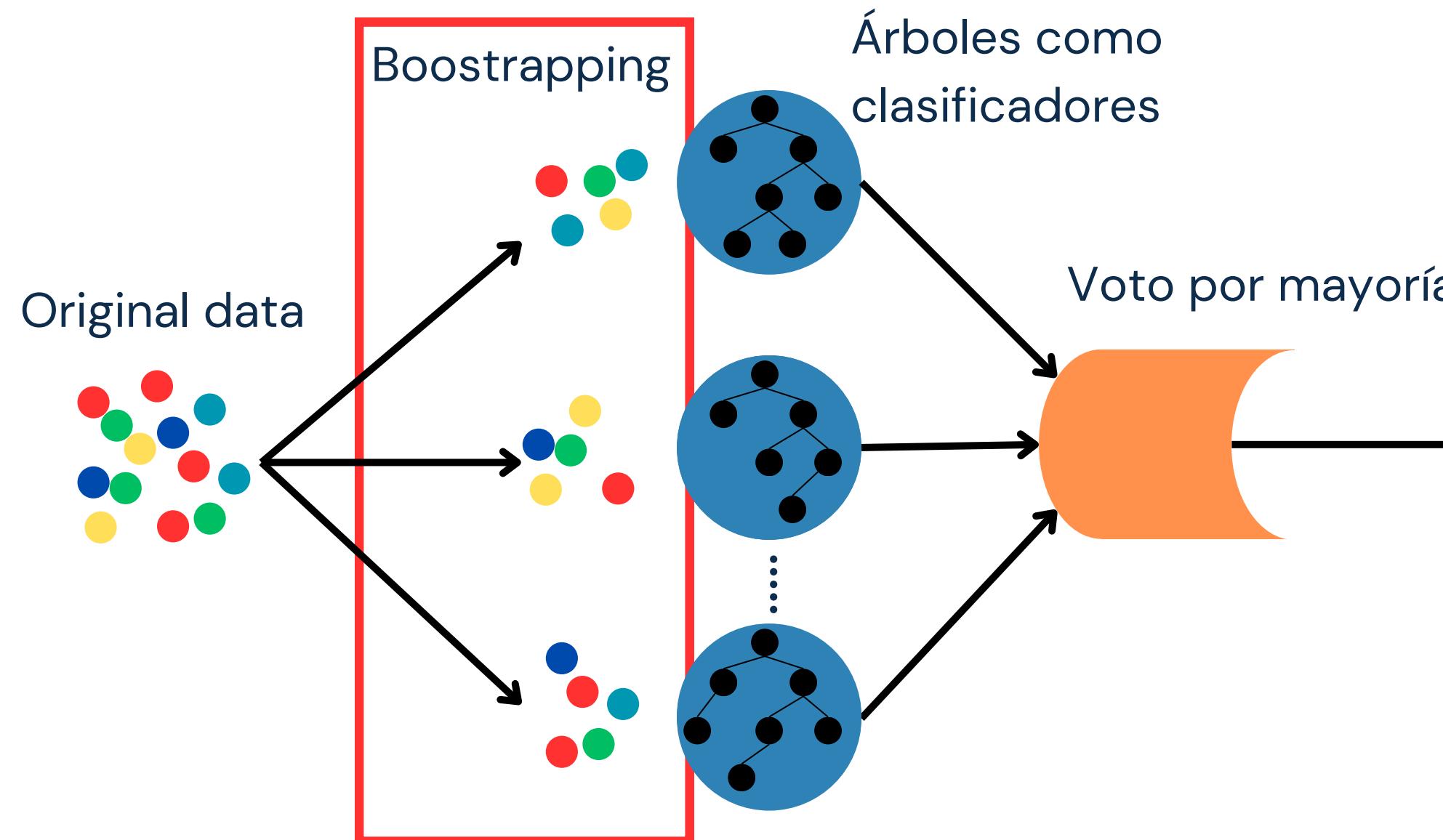
6 - Evaluar todas las divisiones posibles sobre estas m variables.

7 - Dividir el nodo usando la mejor variable y su valor óptimo de división (métrica Gini o ganancia de información).

8 end

Random Forest

Breiman 2001

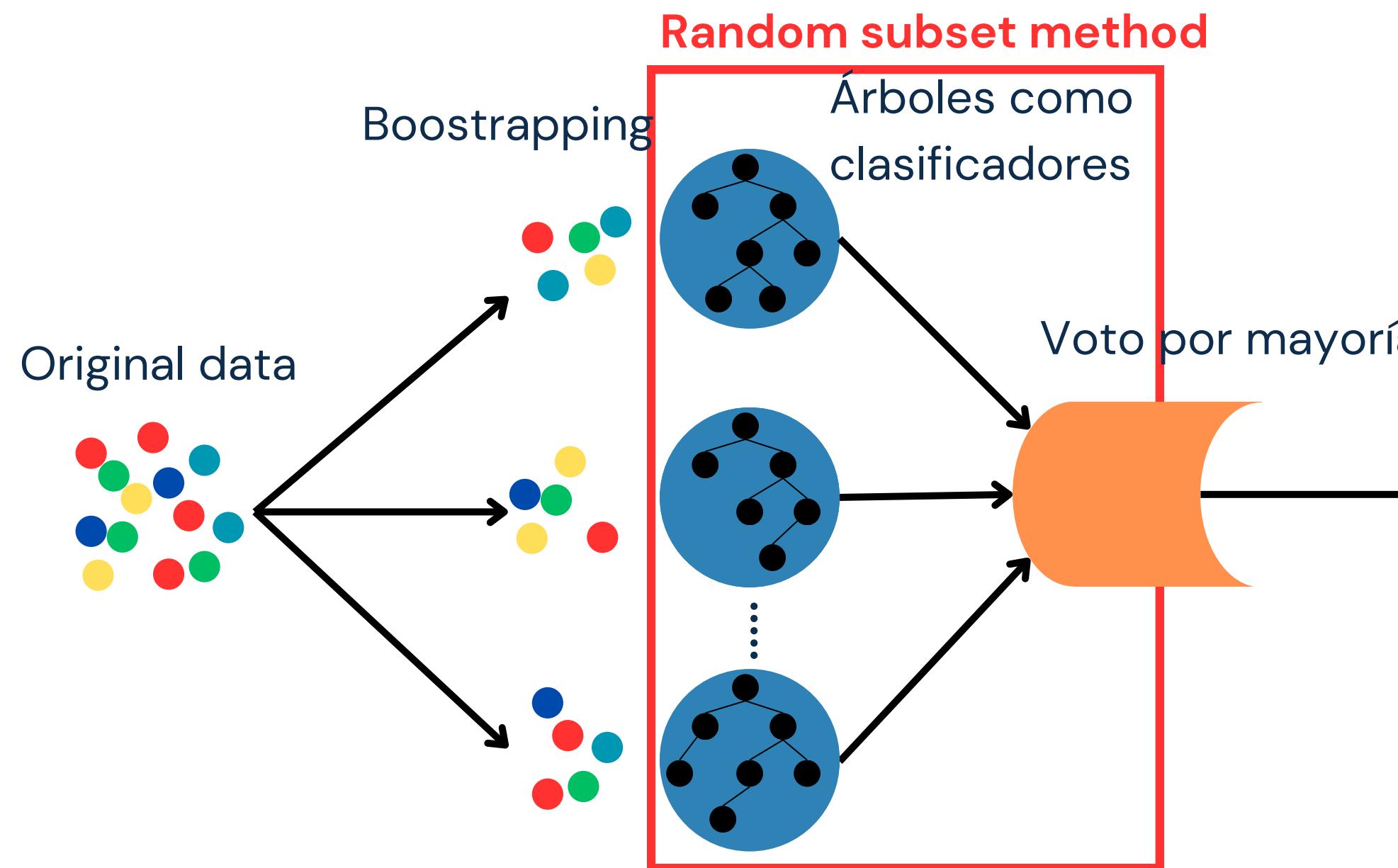


1. Muestreo Bootstrap: Muestreo aleatorio con reemplazo de las instancias.

- Cada árbol se entrena con un subconjunto diferente del conjunto de datos original.
- Reducción de riesgo de sobreajuste.

Random Forest

Breiman 2001



2.1. Aleatoriedad en las variables: selección aleatoria de características en cada nodo del árbol.

- Se consideran solamente algunas características (m) de tamaño fijo para encontrar la mejor división. Donde:

$$m = \sqrt{M}$$

Siendo M el número total de características.

- Se divide el nodo usando la mejor variable y su valor óptimo de división.

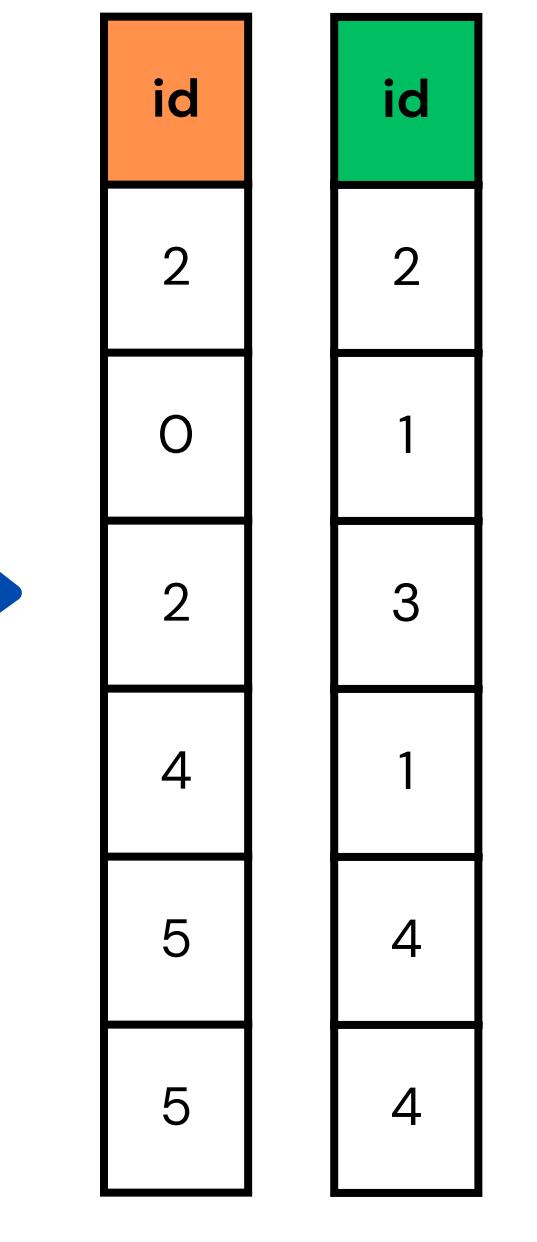
Random Forest

Breiman 2001

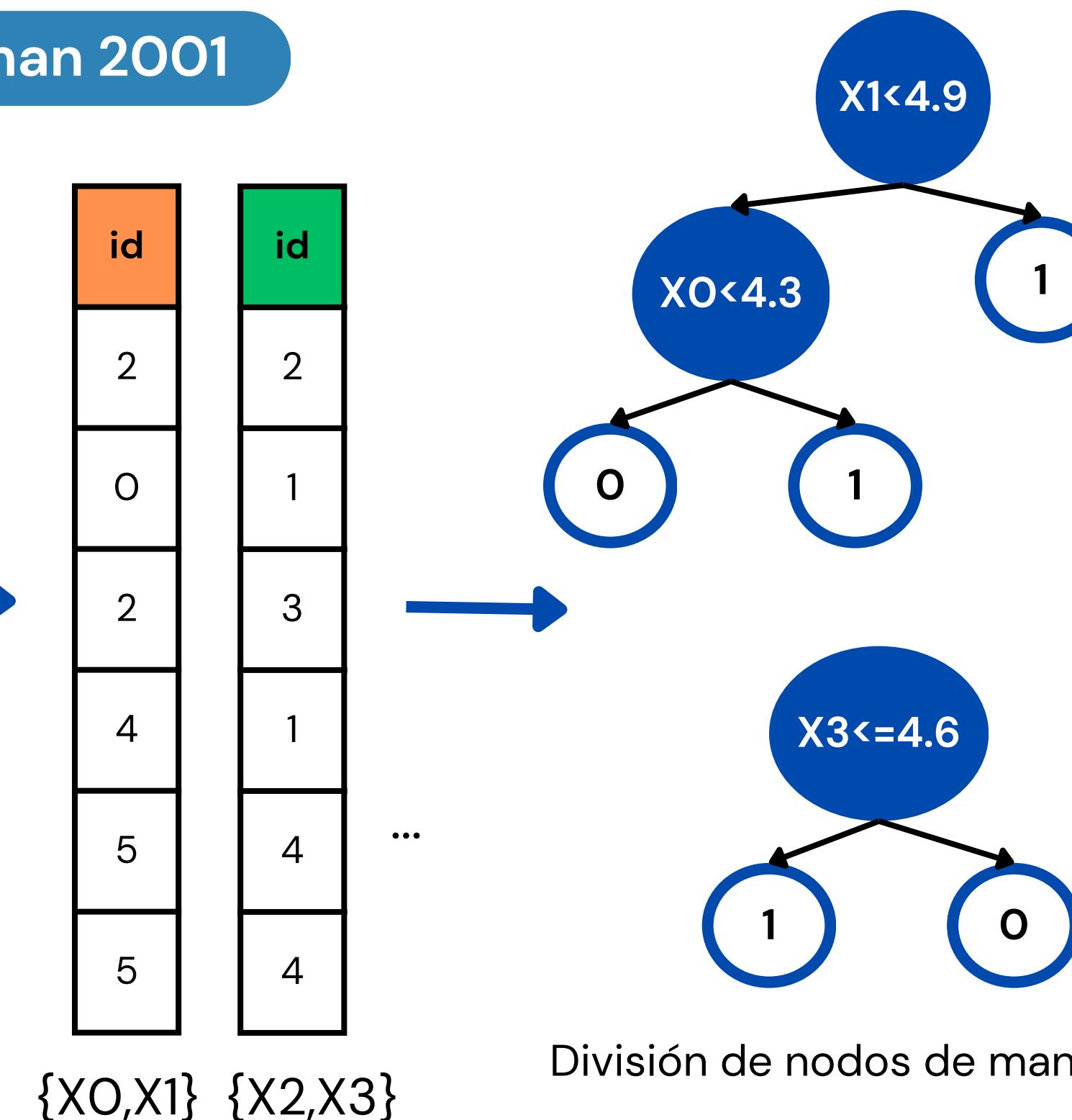
N

id	X0	X1	X2	X3	X4	y
0	4.3	4.9	4.7	4.7	5.5	0
1	3.9	6.1	5.9	5.5	5.9	0
2	3.7	4.8	4.1	5.0	5.6	0
3	6.6	4.4	4.5	3.9	5.9	1
4	6.5	2.9	4.7	4.6	6.1	1
5	2.7	6.7	4.2	5.3	4.8	1

M

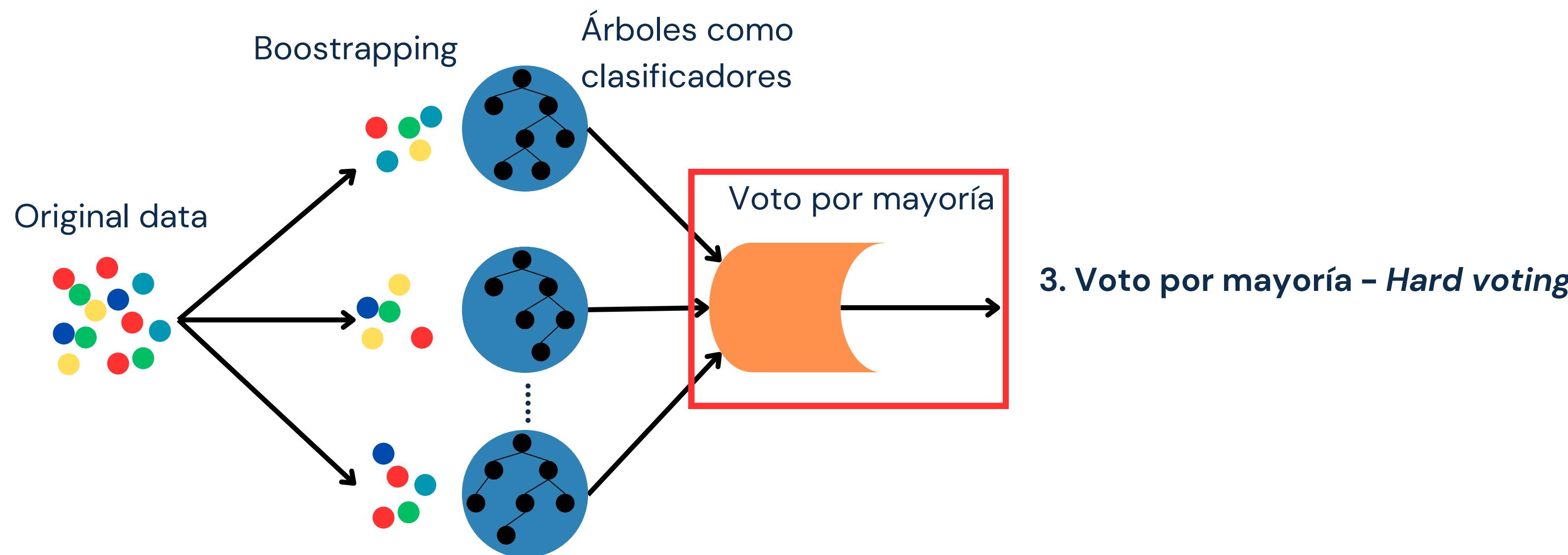


Breiman 2001



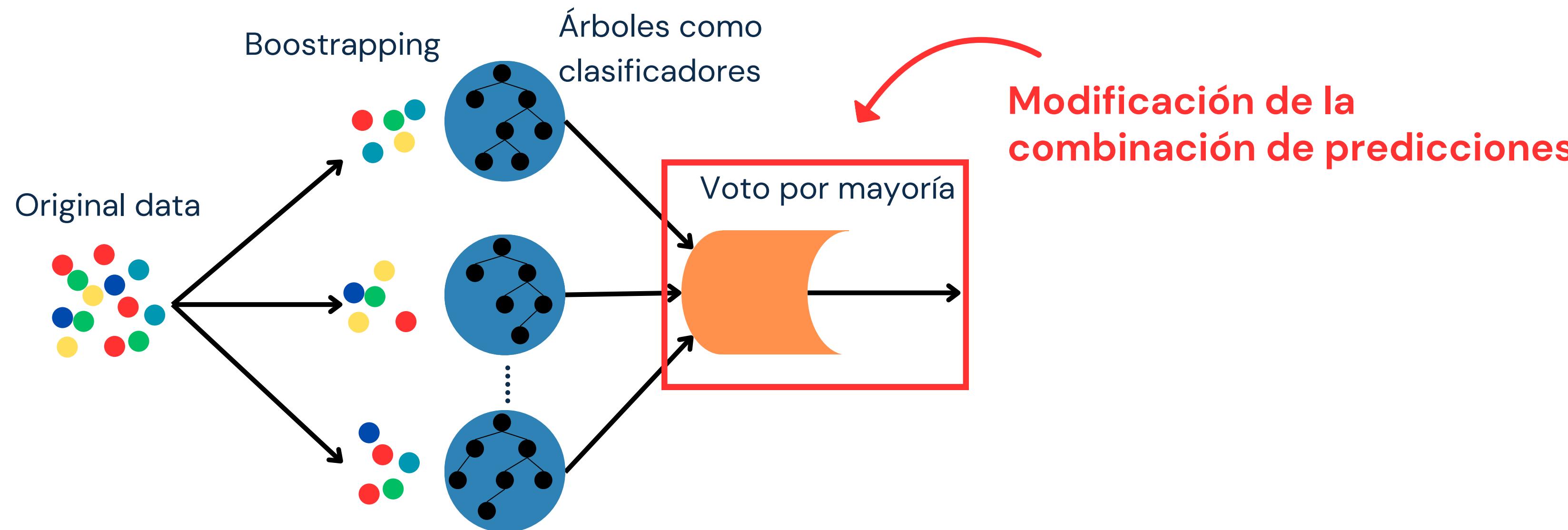
Random Forest

Breiman 2001



Variantes Random Forest

En función de la parte modificada del algoritmo



Variantes Random Forest

Modificación de la combinación de las predicciones de los árboles

Diferentes mecanismos de votación

Tripoliti et al. 2013

El procedimiento de votación se expresa en términos de **asignar pesos** a los votos de los árboles o **seleccionar un subconjunto de árboles** del bosque que sean **más predictivos y menos correlacionados**.

Para este propósito, se emplearon procedimientos de asignación de pesos a los votos de los árboles, selección de características y técnicas de agrupamiento (clustering).

Variantes Random Forest

Modificación de la combinación de las predicciones de los árboles

Potential Nearest Neighbor

Li et al. 2018

El nuevo mecanismo de votación utiliza la **frecuencia de aparición de los vecinos potenciales más cercanos** en los árboles del bosque para identificar al vecino más representativo.

La clase de este vecino determina la predicción final del modelo.

Esto mejora la precisión en comparación con la votación por mayoría simple al aprovechar mejor la información acumulada en los árboles del bosque.

$$f(x_t) = \text{vote} \left(\arg \max_{x_i \in S} Fr(x_i) \right)$$

Siendo $Fr(x_i)$ la frecuencia de cada vecino.

Variantes Random Forest

En función de la parte modificada del algoritmo

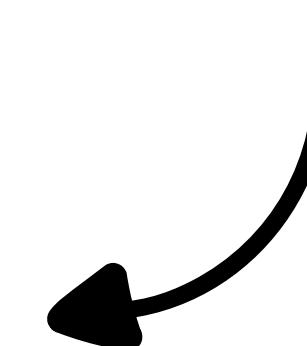
Algorithm 2: Random Forest para Clasificación

Input: N(número de casos prueba), M(número de variables), B(número de árboles)

Output: Conjunto de árboles de decisión $\{T_b\}_1^B$

- 1 $m = \sqrt{M}$
 - 2 for $b=1, \dots, B$ do
 - 3 (i) Extraer una muestra de bootstrap de tamaño N (con reemplazo) del conjunto de entrenamiento. Usar las instancias no seleccionadas (*out-of-bag*) para evaluar el error.
 - 4 (ii) Crecer un árbol de decisión binario T_b para los datos de bootstrap de la siguiente manera, hasta que se cumpla el criterio de detención:
 - 5 - Seleccionar aleatoriamente m variables del conjunto de características.
 - 6 - Evaluar todas las divisiones posibles sobre estas m variables.
 - 7 - Dividir el nodo usando la mejor variable y su valor óptimo de división (métrica Gini o ganancia de información).
 - 8 end
-

Modificación en los criterios de división de nodos



Variantes Random Forest

Modificación en los criterios de división de nodos

Extremely Randomized Trees (Extra-Trees)

Geurts et al. 2008

- Añade aleatorización.
 - En lugar de buscar la mejor división de características, las divisiones en cada nodo se seleccionan aleatoriamente.
- Utiliza todo el conjunto de entrenamiento en lugar de muestras bootstrap, lo que reduce el sesgo.

Ventajas:

- Mayor velocidad de entrenamiento:
 - Las divisiones de los nodos no necesitan ser optimizadas; los puntos de corte se seleccionan aleatoriamente, lo que reduce el tiempo de cómputo.
- Menor riesgo de sobreajuste:
 - La alta aleatoriedad en las divisiones hace que los árboles individuales sean más variados, lo que mejora la capacidad de generalización del modelo.
- Adecuado para grandes conjuntos de datos:

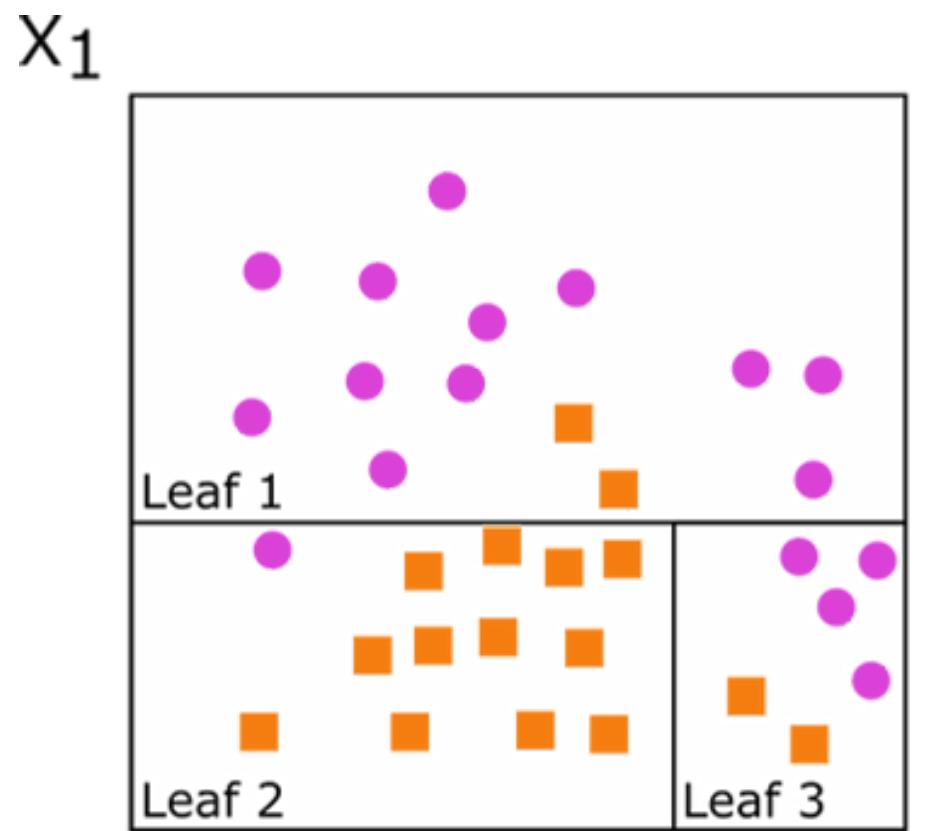
Variantes Random Forest

Modificación en los criterios de división de nodos

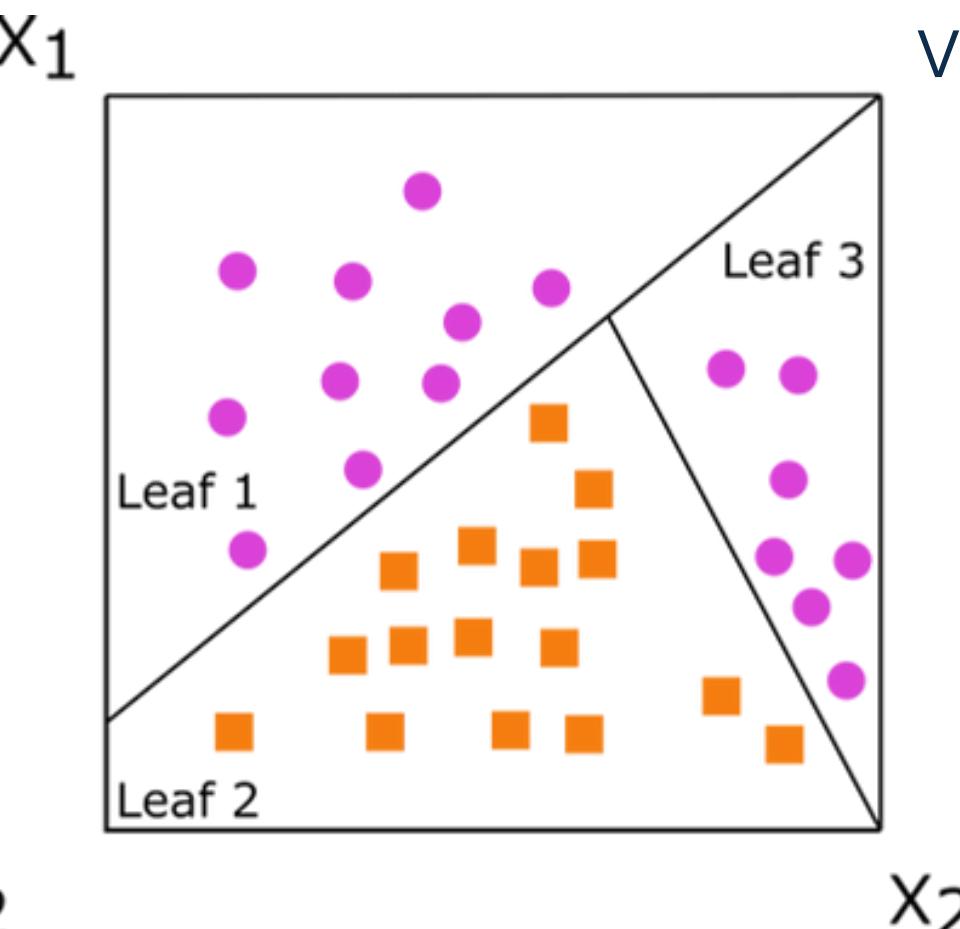
Oblique Random Forest

Menze et al. 2009

- Uso de hiperplanos oblicuos
 - Divisiones más complejas y ajustadas, para combinar características.



[Link imagen](#)



Ventajas:

- Reducción en el número de nodos:
 - Al realizar divisiones más eficientes, el modelo puede necesitar menos nodos para separar correctamente los datos.
- Generalización mejorada:
 - Las divisiones oblicuas reducen el riesgo de sobreajuste en comparación con las divisiones paralelas a los ejes.

Variantes Random Forest

En función de la parte modificada del algoritmo

Algorithm 2: Random Forest para Clasificación

Input: N(número de casos prueba), M(número de variables), B(número de árboles)

Output: Conjunto de árboles de decisión $\{T_b\}_1^B$

```

1  $m = \sqrt{M}$ 
2 for b=1, ..., B do
3   (i) Extraer una muestra de bootstrap de tamaño  $N$  (con reemplazo) del conjunto de
      entrenamiento. Usar las instancias no seleccionadas (out-of-bag) para evaluar el error.
4   (ii) Crecer un árbol de decisión binario  $T_b$  para los datos de bootstrap de la siguiente
      manera, hasta que se cumpla el criterio de detención:
5     - Seleccionar aleatoriamente  $m$  variables del conjunto de características.
6     - Evaluar todas las divisiones posibles sobre estas  $m$  variables.
7     - Dividir el nodo usando la mejor variable y su valor óptimo de división (métrica Gini o
      ganancia de información).
8 end

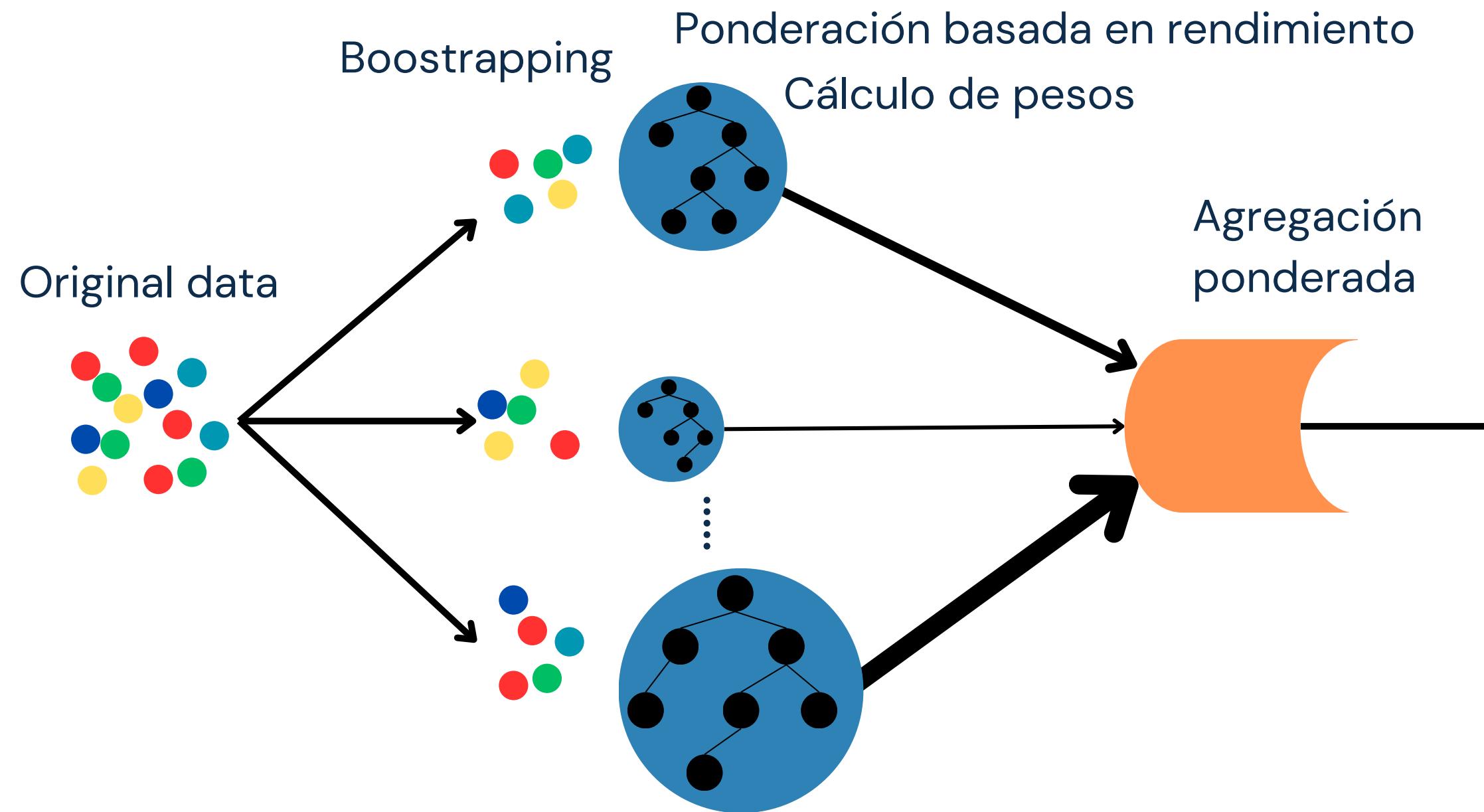
```

**Modificación en la
construcción del árbol**

Variantes Random Forest

Modificación en la construcción del árbol

Weighted Random Forest (wRF) Chen et al. 2004



- Diseñado para abordar problemas en datos de alta dimensionalidad.
- Se asigna **pesos** diferentes a cada árbol en función de su rendimiento individual.
- Los más precisos reciben mayor peso durante la agregación de predicciones.
- Complejidad en entrenamiento.

Presentador 4



POLITÉCNICA

UNIVERSIDAD
POLÍTÉCNICA
DE MADRID

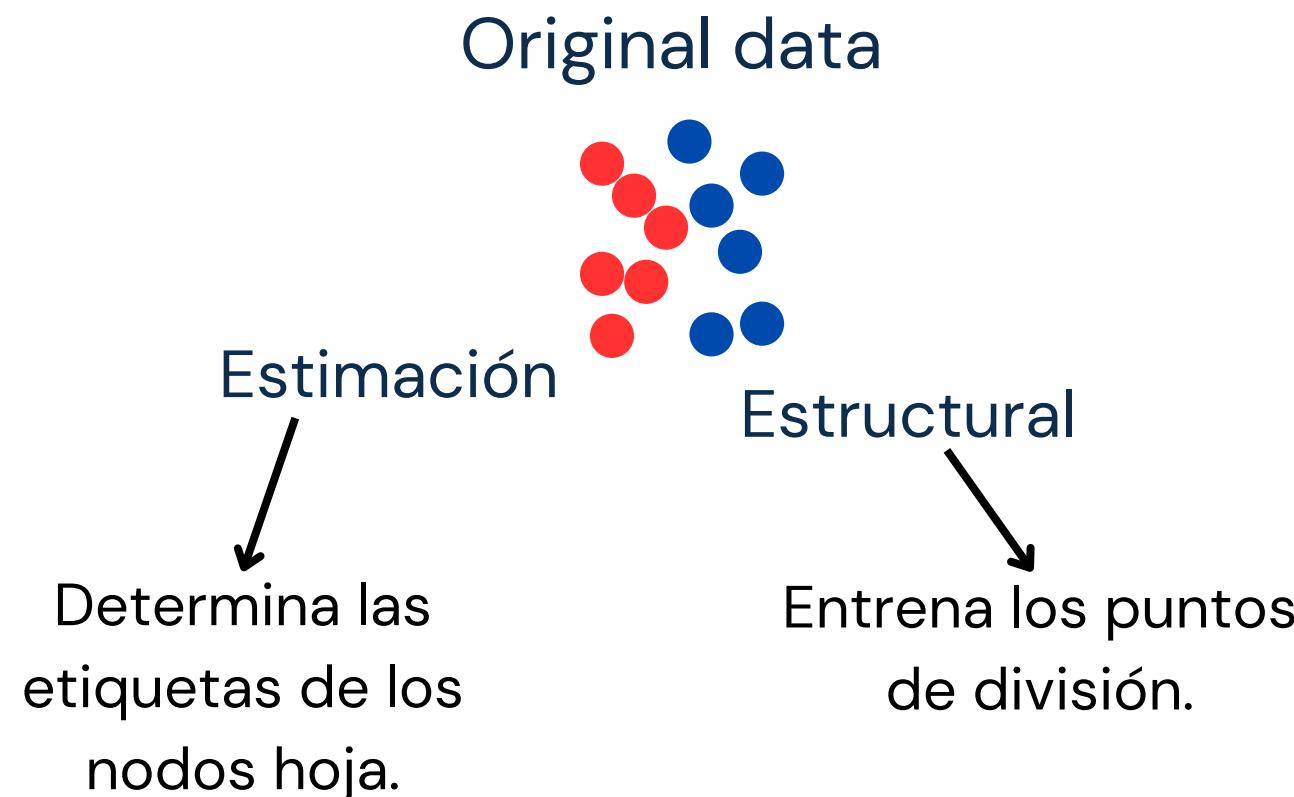


ESCUELA TÉCNICA
SUPERIOR DE INGENIEROS
INFORMÁTICOS

Variantes Random Forest

Denil14

Denil et al. 2014

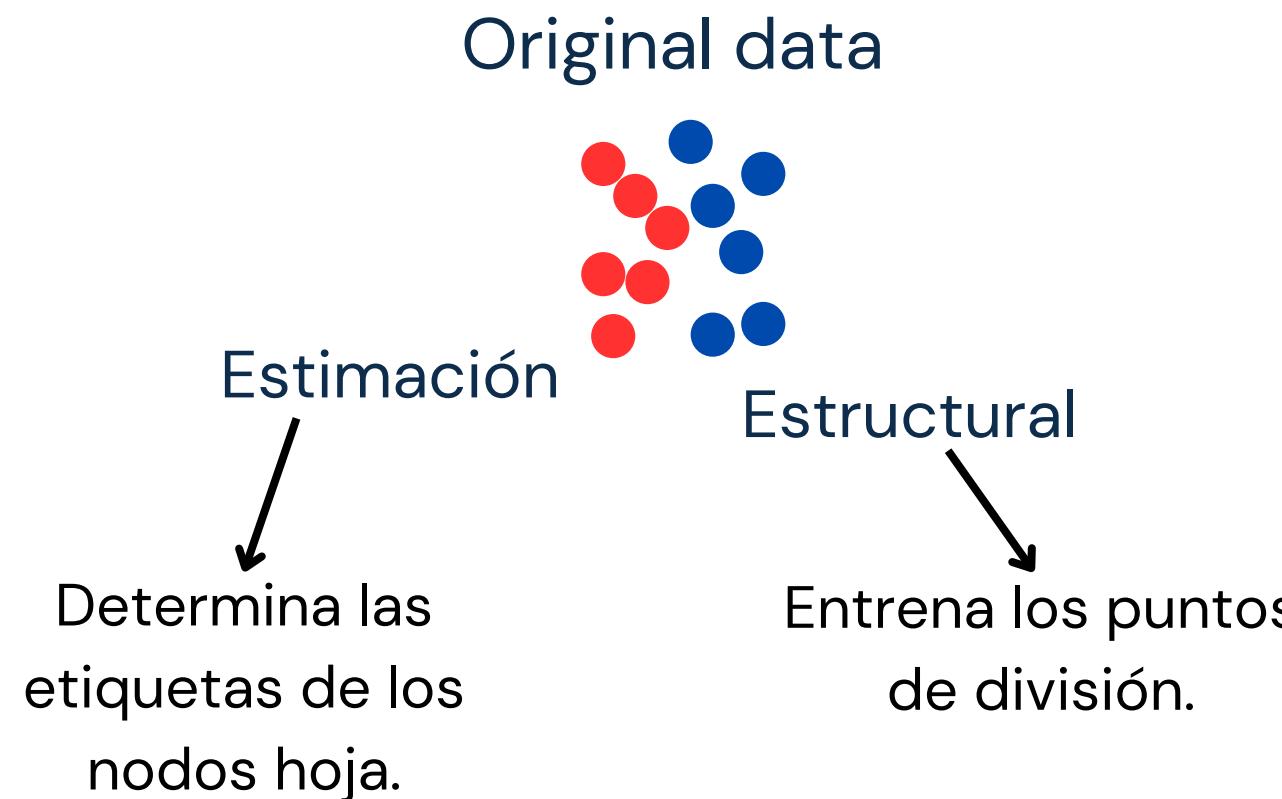


- **División del conjunto de datos:**
 - Parte estructural: Entrena los puntos de división.
 - Parte de estimación: Determina las etiquetas de los nodos hoja.
- **Selección de características:**
 - Tamaño del subespacio de características basado en una **distribución de Poisson**.
- **Optimización:**
 - Se buscan la característica y el valor de división óptimos utilizando muestras preseleccionadas de la parte estructural.

Variantes Random Forest

Bernoulli RF (BRF)

Wang et al. 2017

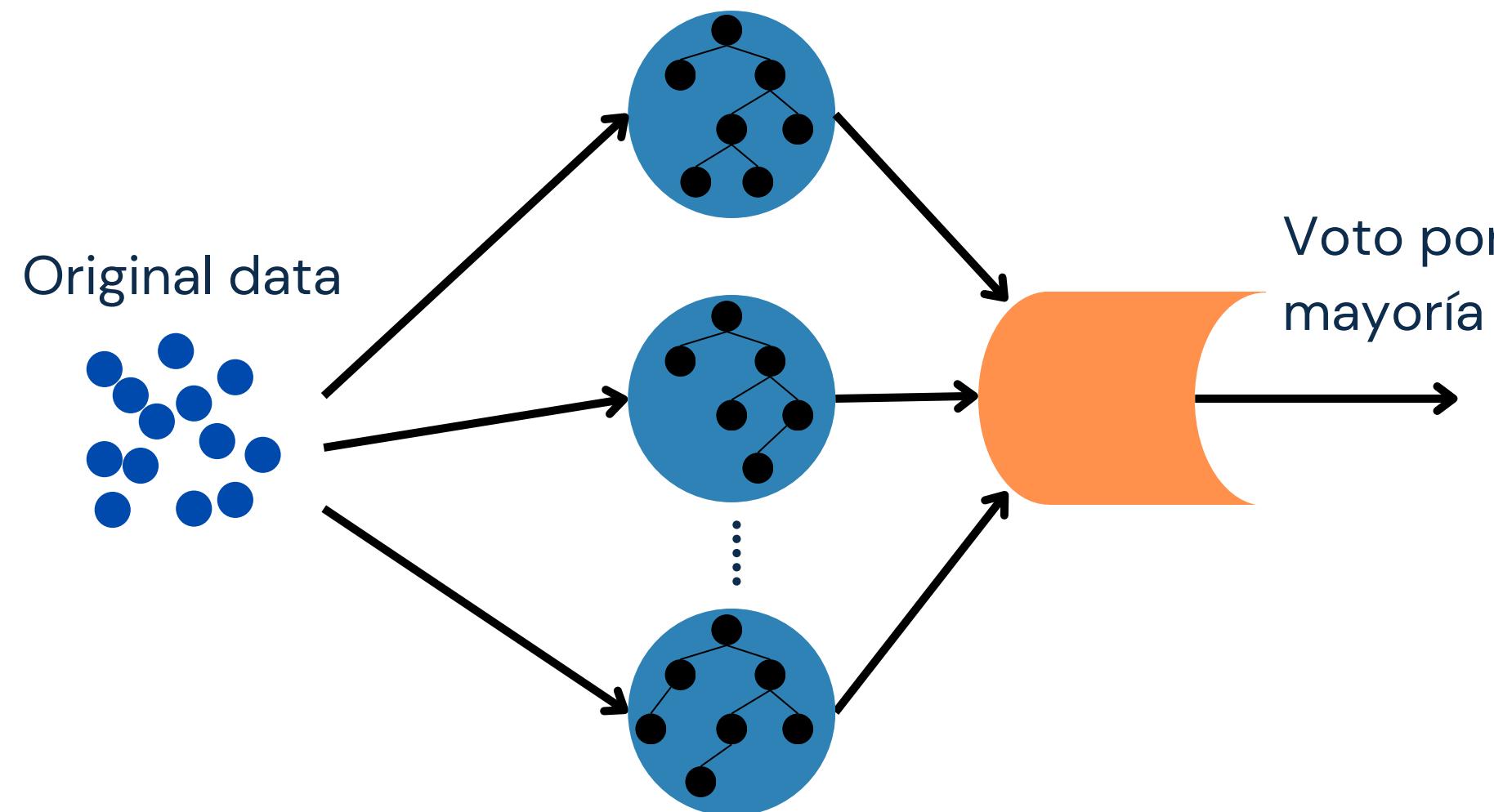


- **Inspirado en Denil14.**
- **2 distribuciones de Bernoulli en cada nodo:**
 - Determina si el tamaño del subespacio de características será o $m = \sqrt{M}$
 - Decide si el valor de división se selecciona aleatoriamente o mediante el criterio óptimo.
- Características:
 - **Consistencia débil.**

Variantes Random Forest

Double Random Forest

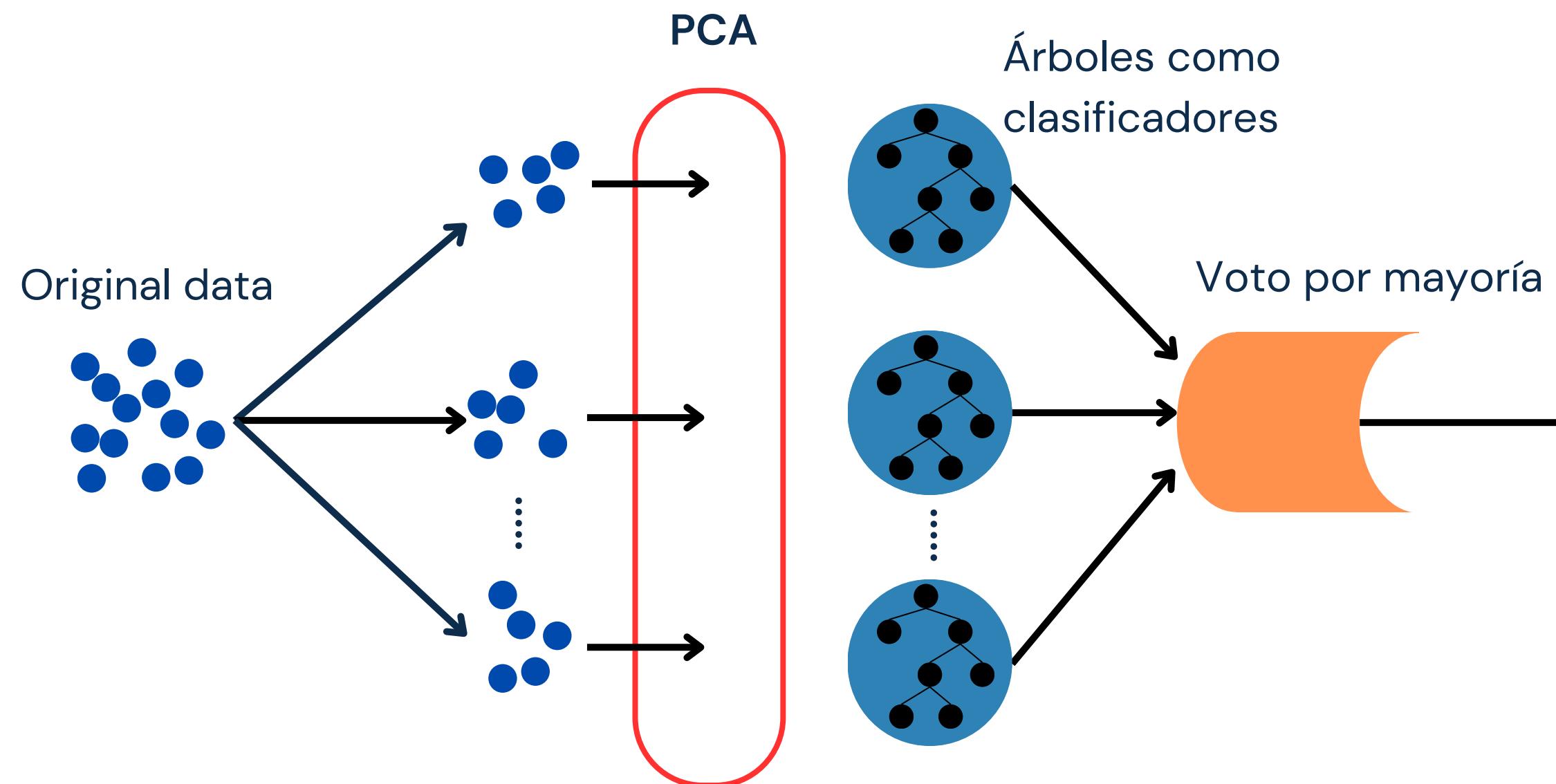
Han et al. 2020



- Cada clasificador base se entrena con el conjunto de **datos original** en lugar de remuestreo completo por bootstrap.
- Utiliza más características únicas, lo que conduce a:
 - Árboles de decisión más grandes.
 - Mejor generalización
- Mecanismo:
 - **El muestreo bootstrap se realiza de manera puntual en cada nodo no terminal**
- Selección aleatoria de características:
 - Al igual que RF, el DRF selecciona aleatoriamente subconjuntos de características en cada nodo para mejorar la diversidad de las particiones.

Rotation Forest

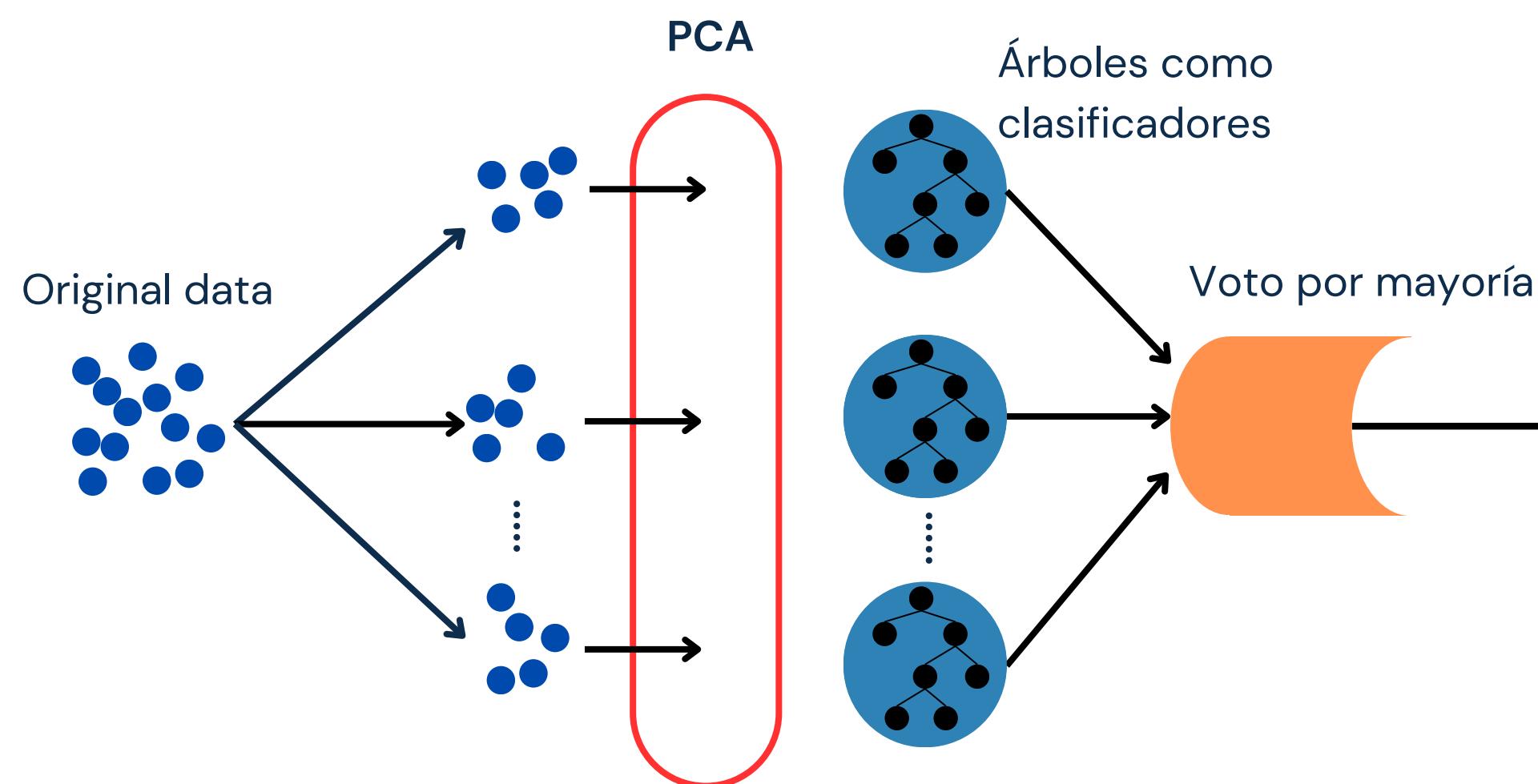
Rodriguez et al. 2006



- Genera diversidad mediante transformaciones lineales aleatorias aplicadas a los datos de entrada.
 - Usa técnicas como el Análisis de **Componentes Principales (PCA)** para rotar los datos en un espacio de características modificado.

Rotation Forest

Rodriguez et al. 2006



Funcionamiento:

- 1. División del conjunto de características:**
 - a. Las características originales se dividen en K subconjuntos aleatorios.
- 2. Transformación por subconjuntos:**
 - a. Para cada subconjunto de características, se realiza un PCA o alguna transformación lineal para rotarlas.
 - b. El PCA genera componentes principales para cada subconjunto, y estas componentes se combinan para formar un nuevo espacio de características rotadas.
- 3. Creación de los clasificadores base (árboles):**
 - a. Cada árbol de decisión del bosque se entrena utilizando el conjunto completo de datos pero en el espacio de características rotadas generado.
- 4. Predicción.**

Variantes Rotation Forest

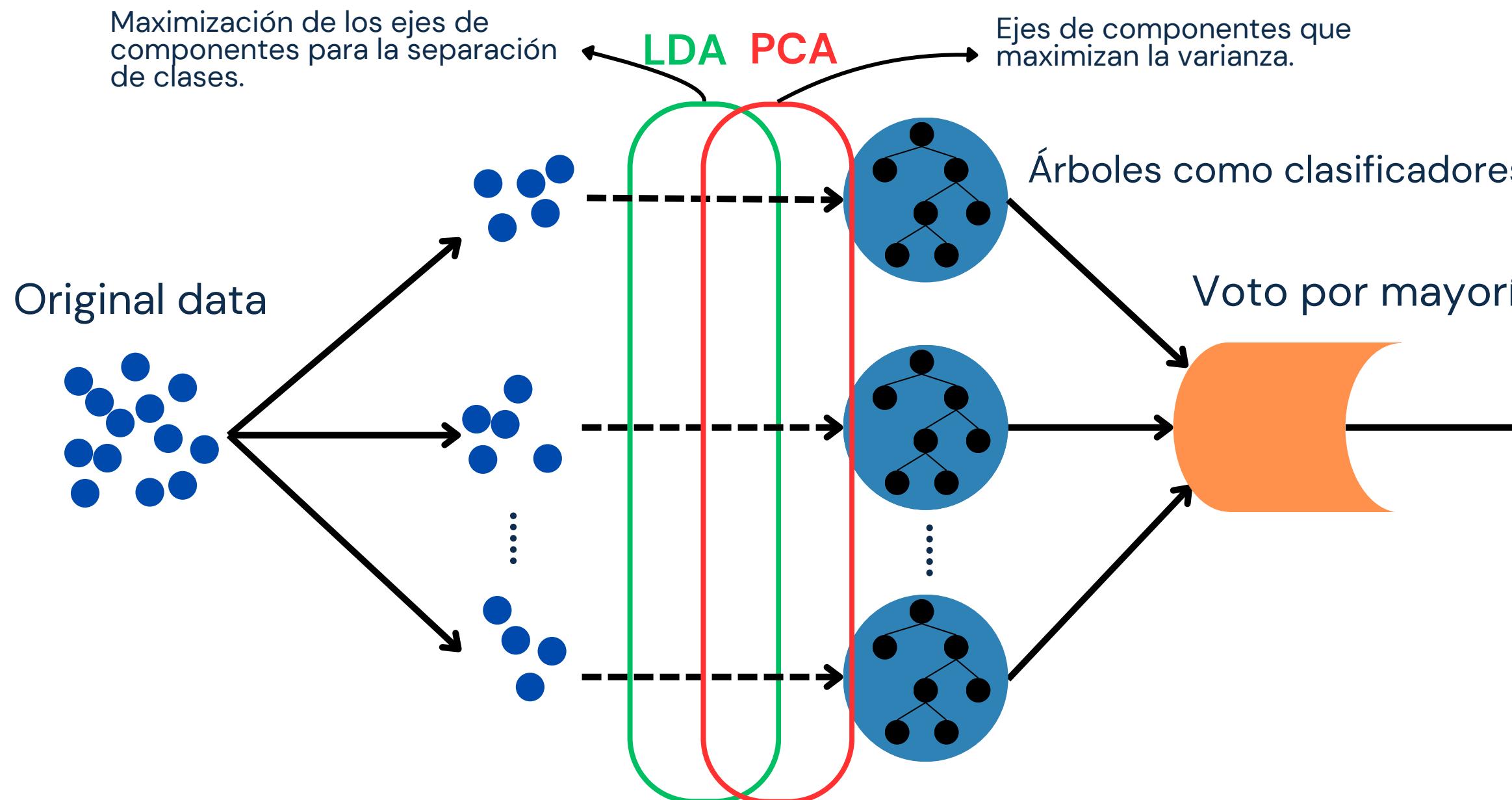
RotBoost

Zhang and Zhang 2010

- Combinación de Rotation Forest y AdaBoost.

Ensemble Feature Forest (FFF)

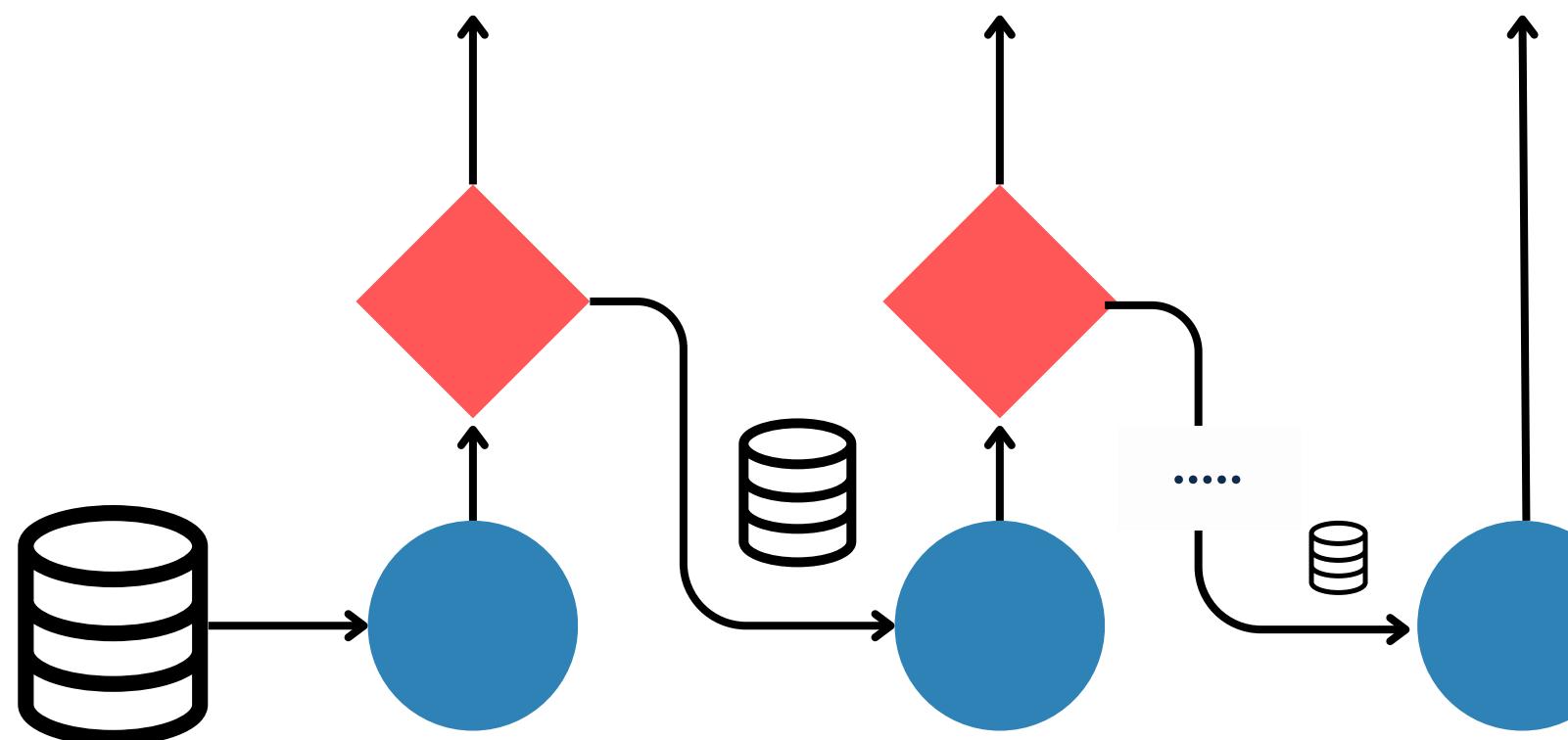
Zhang and Suganthan 2014



- Entrenamiento con PCA y LDA.
- Posteriormente, se concatena las características proyectadas junto con las características originales en un espacio de mayor dimensionalidad en **nodo raíz**.

Cascading

Alpaydin and Kaynak 1997



Procesan las instancias de manera progresiva, pasando por múltiples etapas de clasificación.

Algorithm 4: Cascading

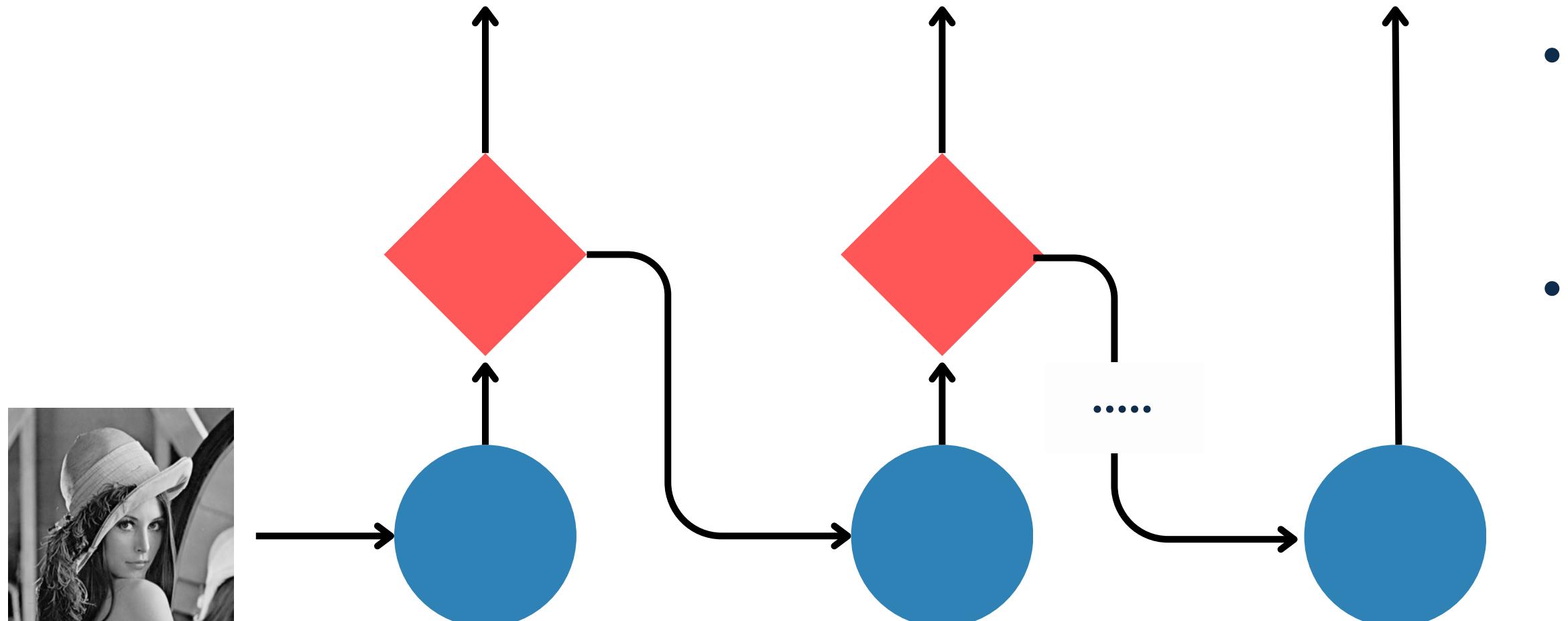
Input: Datos de entrenamiento \mathcal{D} , número de niveles de la cascada L , modelos base $\{\phi_\ell\}_{\ell=1}^L$, umbral de confianza τ_ℓ para cada nivel ℓ

Output: Conjunto de modelos $\{\phi_\ell\}_{\ell=1}^L$ y predicción final

- 1 **for** $\ell = 1, \dots, L$ **do**
- 2 **if** $\ell = 1$ **then**
 - Entrenar ϕ_1 con el conjunto completo $\mathcal{D}_1 = \mathcal{D}$;
- 3 **else**
 - Seleccionar las instancias de \mathcal{D} no clasificadas con confianza por los niveles anteriores:
- 4 $\mathcal{D}_\ell = \{(x_i, y_i) \in \mathcal{D} : P(y_i | \phi_{\ell-1}(x_i)) < \tau_{\ell-1} \text{ o } \phi_{\ell-1}(x_i) \neq y_i\}$.
- 5 - Entrenar ϕ_ℓ con \mathcal{D}_ℓ ;
- 6 **Predicción:** Para una nueva instancia x : **for** $\ell = 1, \dots, L$ **do**
 - Calcular la predicción $\phi_\ell(x)$ con su confianza $P(y | \phi_\ell(x))$;
 - **if** $P(y | \phi_\ell(x)) \geq \tau_\ell$ **then**
 - Retornar $\phi_\ell(x)$ como la predicción final;
- 7 **Si ningún nivel clasifica con confianza:** Usar un clasificador no paramétrico (por ejemplo, k -NN) para manejar las instancias restantes.

Variantes Cascading

Cascading Harr



$\theta_0 \ \theta_1 \ \theta_2 \ \dots$



Viola & Jones 2001

Características

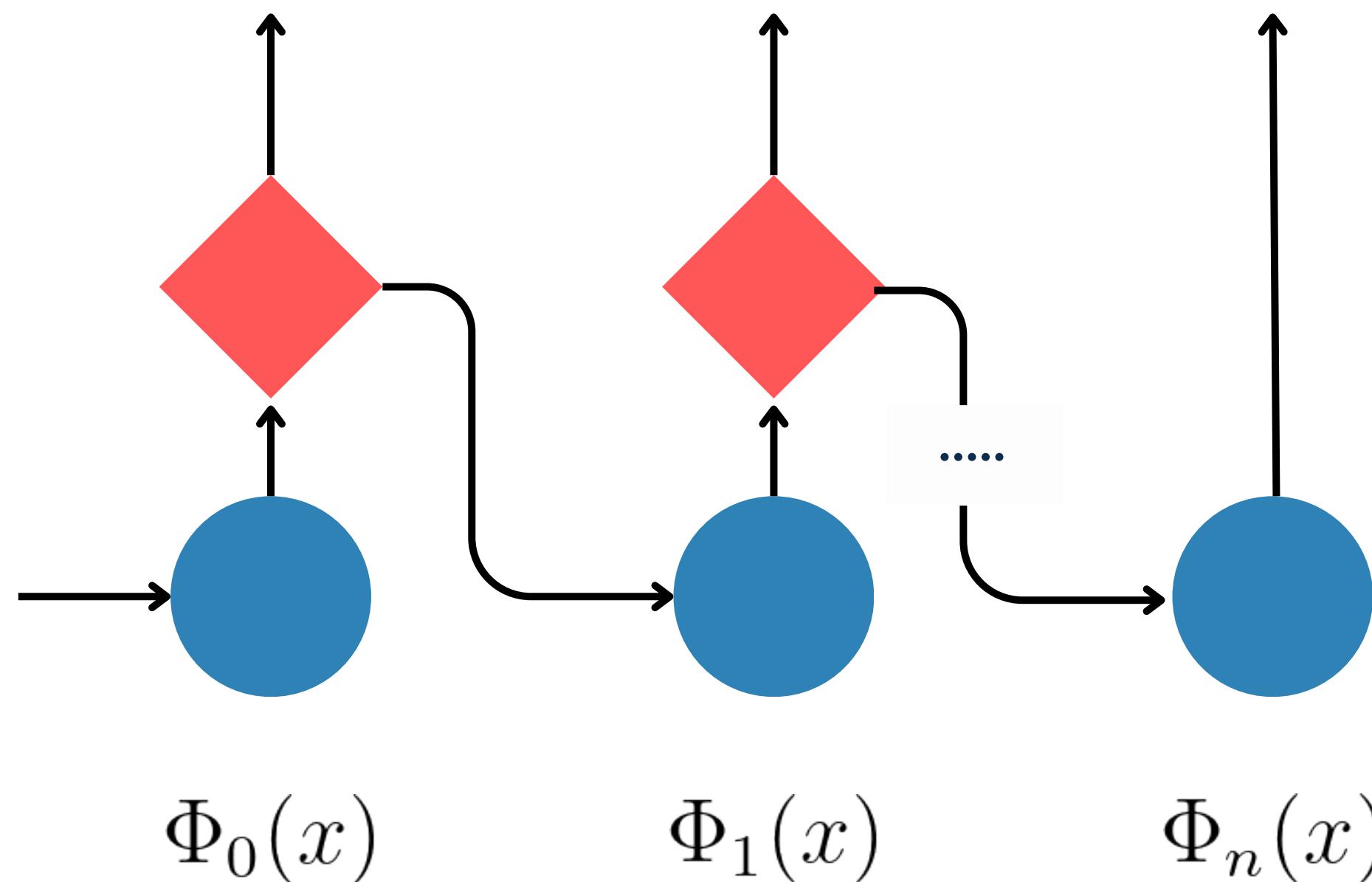
- Hace uso de imágenes integrales y selección de características mediante AdaBoost.
- Clasificadores débiles para cada característica

$$h_j(x) = \begin{cases} 1, & p_j f_j(x) < p_j \theta_j, \\ 0, & \text{en otro caso,} \end{cases}$$

Variantes Cascading

Boosting Chain

Xiao y Zhu 2003



Cada clasificador de la cadena utiliza las predicciones acumuladas de los clasificadores anteriores como base para su entrenamiento.

$$\Phi_i(x) = \text{sign} \left(\sum_{k=1}^{i-1} \sum_{t=0}^{m_k} \alpha_{t,k} h_{t,k}(x) - b_k \right)$$

Características

- Ajusta dinámicamente los pesos para enfocar en áreas mas difíciles.
- Mejora de entrenamiento debido al paso de información entre niveles.

Presentador 5



POLITÉCNICA

UNIVERSIDAD
POLITÉCNICA
DE MADRID

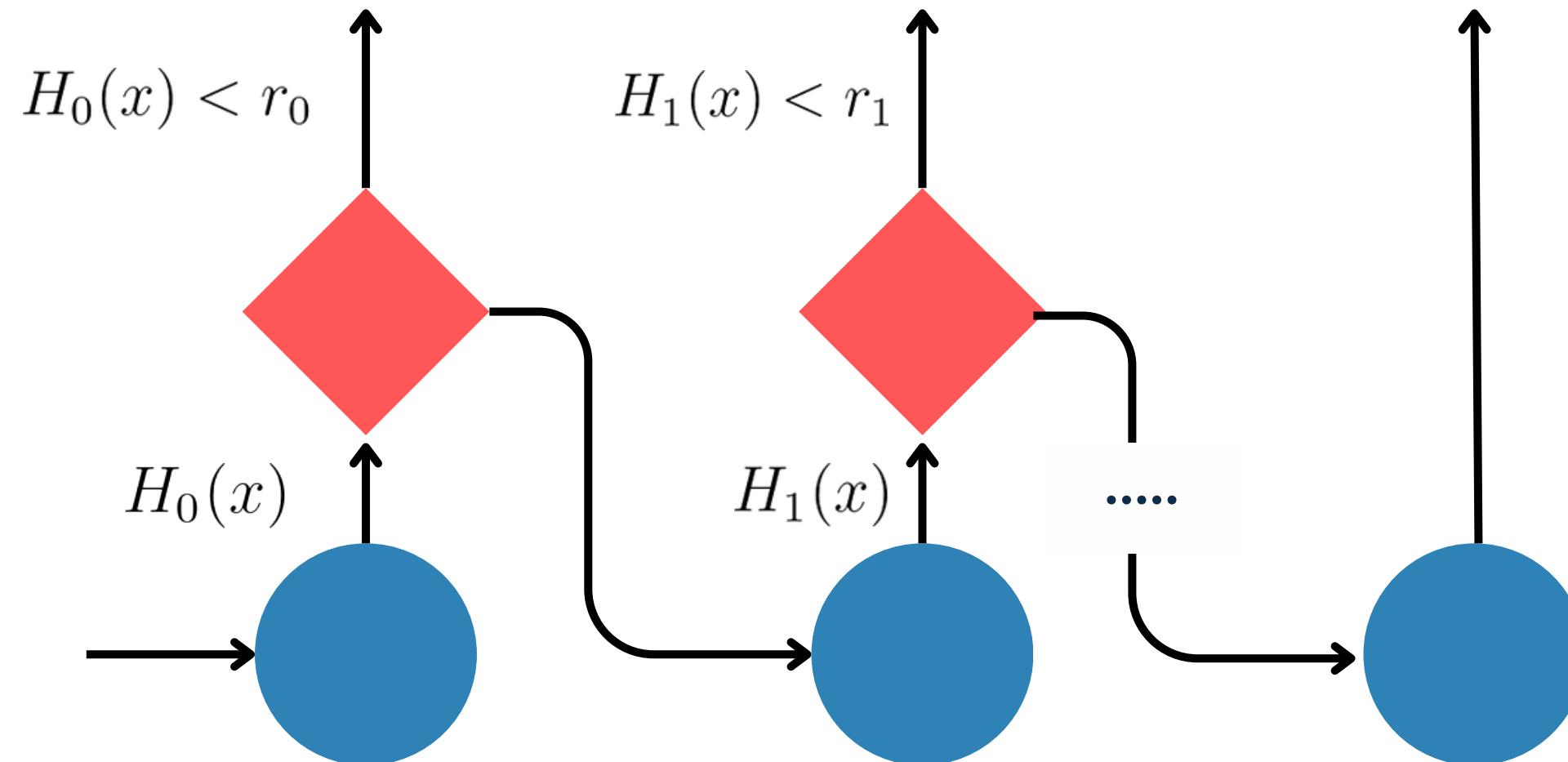


ESCUELA TÉCNICA
SUPERIOR DE INGENIEROS
INFORMÁTICOS

Variantes Cascading

Soft Cascade

Bourdev y Brandt 2005



Características

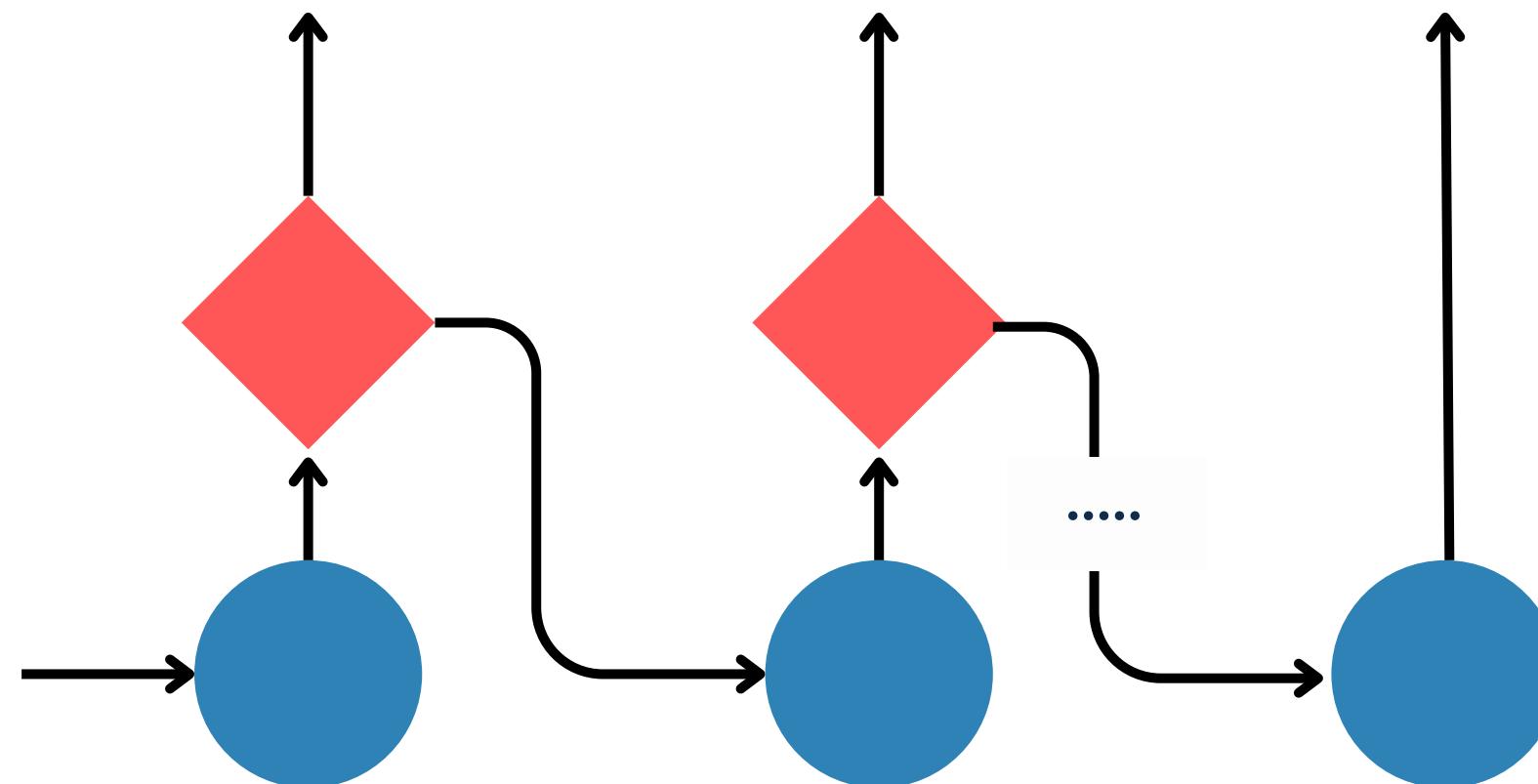
- Hace uso de umbrales acumulativos
- Entrenamiento del modelo y calibración de umbrales se hacen de forma independiente

Actualización automática umbral

- Se explora la superficie ROC para evaluar configuraciones y seleccionar las más eficientes

Variantes Cascading

FCBoost



Saberian and Vasconcelos 2010

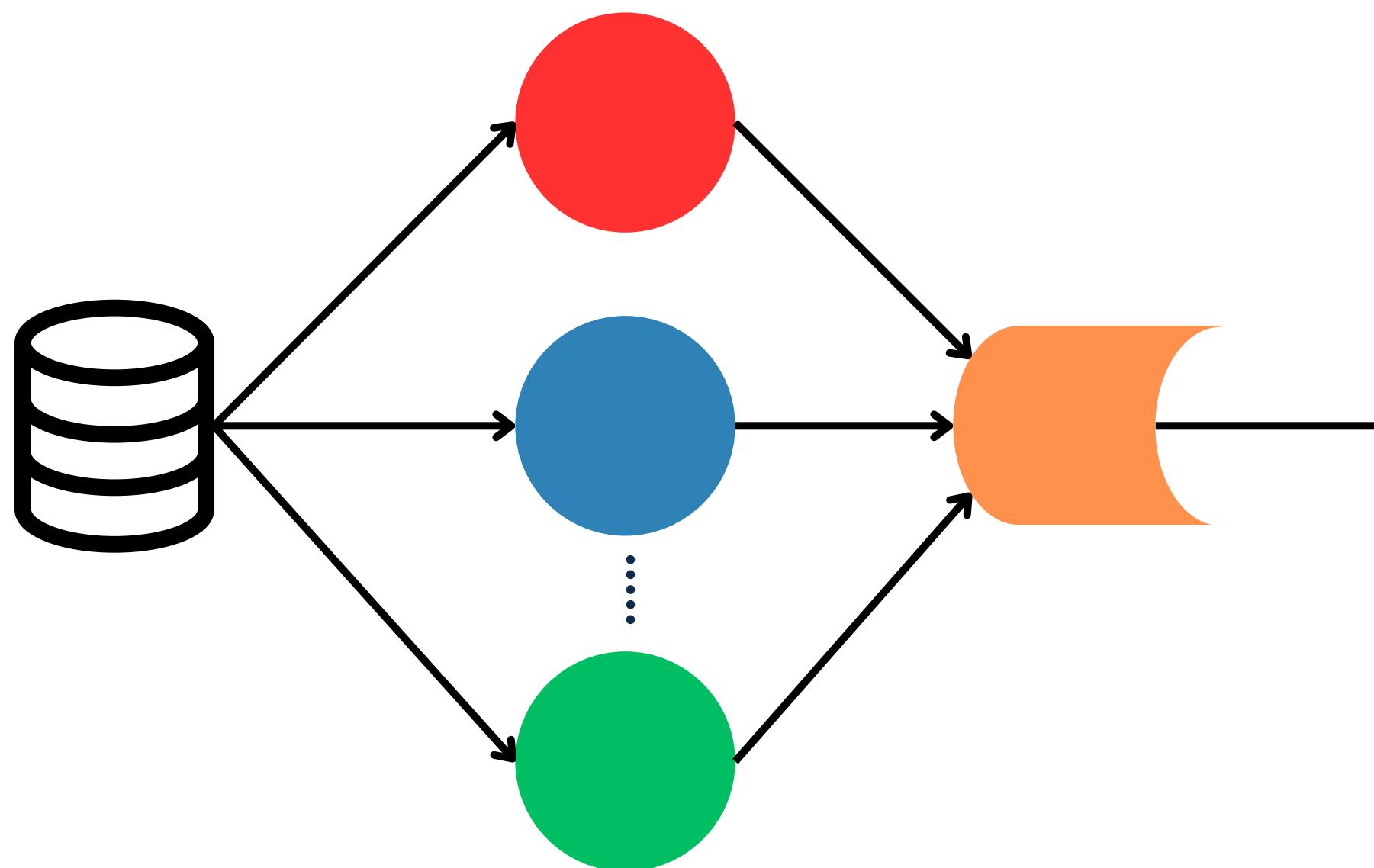
Se caracteriza por la **actualización parámetros automática**

- Número de etapas: Determinado según su contribución al riesgo total del modelo.
- Clasificadores débiles por etapa: Calculados iterativamente para equilibrar precisión y eficiencia.
- Umbrales de rechazo: Optimizados conjuntamente con el entrenamiento para maximizar la eficacia.

Contribución acumulativa: Coeficientes específicos que ajustan el impacto de cada etapa en la decisión final.

Bootstrapping dinámico: Actualización continua del conjunto de entrenamiento mediante ejemplos falsos positivos.

Clasificadores Híbridos



- Conjunto de clasificadores heterogéneos.
- Combinar múltiples clasificadores individuales de diferentes tipos.
- Mejora del rendimiento general.
- Aprovechar virtudes de los diferentes algoritmos base.

Clasificadores Híbridos

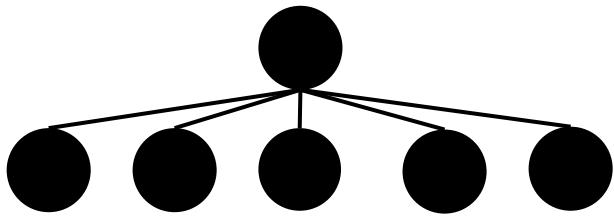
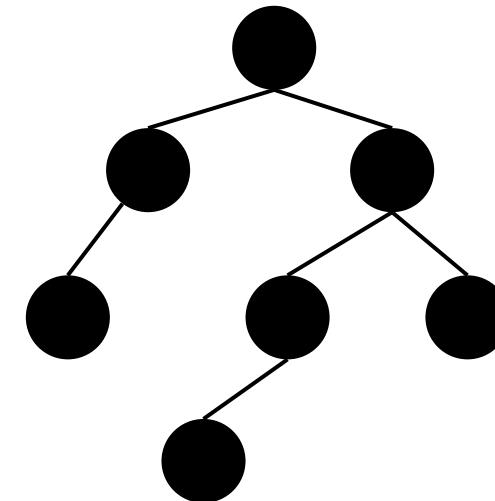
NBTree

Kohavi 1996

Árbol de decisión



Naive Bayes



Acumulación de evidencias de múltiples atributos para aumentar la precisión.

Segmentación de los datos para facilitar la clasificación

Funcionamiento del algoritmo

- Evaluación de la utilidad de la división
- Selección del mejor atributo
- División recursiva
- Creación de clasificadores Naïve Bayes

Clasificadores Híbridos

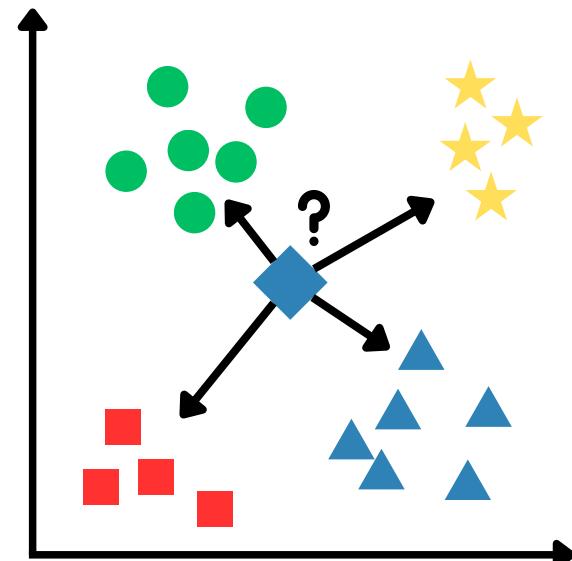
Lazy Bayesian Rules

Zheng and Webb 2000

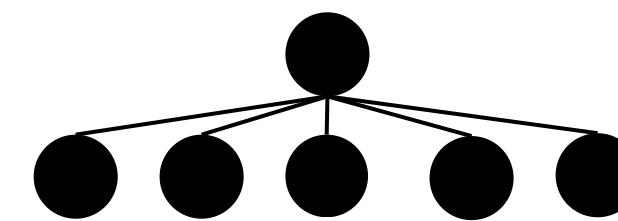
Lazy learning



Naive Bayes



Construye un modelo específico en el momento de clasificar una nueva instancia.



Aprovecha la independencia condicional de los atributos para construir un modelo probabilístico.

Claves:

- Usar únicamente las instancias más relevantes para cada caso
- Construir una regla bayesiana personalizada para cada instancia

Funcionamiento del algoritmo

- Inicialización
- Iteración sobre los atributos
- Finalización
- Clasificación

Clasificadores Híbridos

Logistic Models Tree

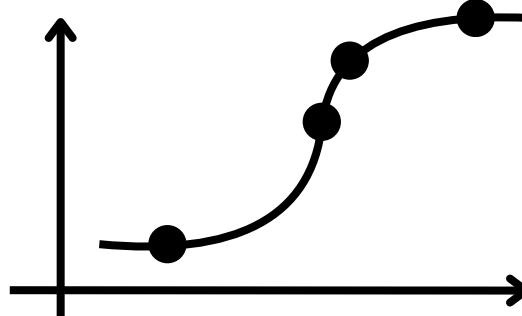
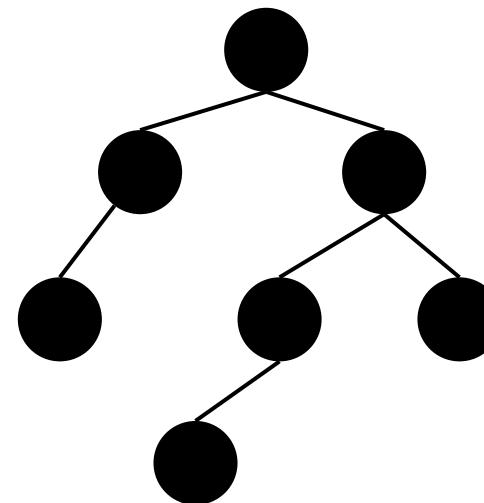
Landwehr 2005

Construcción de un árbol de decisión en el que cada hoja contiene un modelo de regresión logística.

Árbol de decisión



Regresión Logística



Robustez para producir modelos predictivos precisos e interpretables

Capacidad de modelar relaciones no lineales en los datos

Funcionamiento del algoritmo

- Crecimiento del árbol
- Construcción de los modelos logísticos
- Poda del árbol

Clasificadores Híbridos

Hybrid Fuzzy Decision Trees

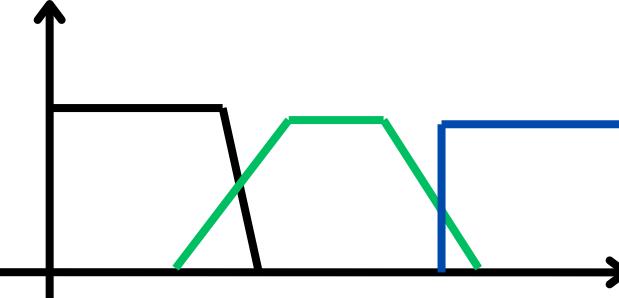
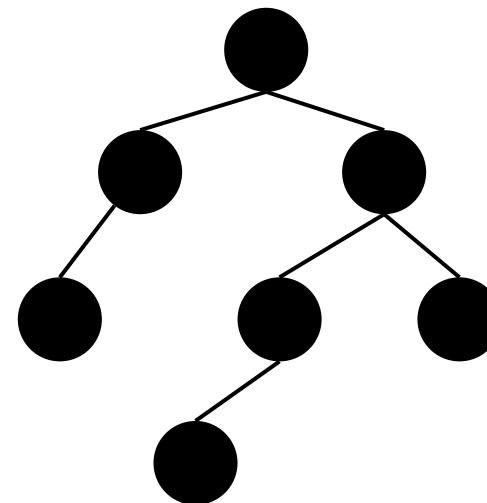
Janikow 1998

Utilizan conjuntos difusos para manejar la incertidumbre en los datos y permiten que las muestras se propaguen por múltiples caminos en el árbol con diferentes grados de confianza.

Árbol de decisión



Lógica difusa



Manejar la incertidumbre en los datos

Segmentación de los datos para facilitar la clasificación

Funcionamiento del algoritmo

- Difusión de los datos
- Construcción del Árbol de Decisión Difuso
- Conversión a Reglas Difusas
- Inferencia Difusa

Conclusiones y líneas futuras

- Bagging y Random Forest:
 - Reducen la varianza y manejan el sobreajuste.
 - Especialmente útiles en datasets grandes y complejos.
 - Boosting:
 - Mejora la precisión al reducir el sesgo.
 - Requiere una configuración cuidadosa para evitar el sobreajuste.
 - Rotation Forest:
 - Utiliza transformaciones del espacio de características para aumentar la diversidad de clasificadores.
 - Optimiza los resultados gracias a su enfoque único.
 - Cascading e Híbridos:
 - Combinan múltiples técnicas para mejorar la precisión, robustez y generalización.
 - Representan la vanguardia en metaclasificadores.
- **Mejorar la transparencia de los metaclasificadores:**
 - Implementar técnicas como explicaciones basadas en importancia, análisis de contribución local o métodos post-hoc como SHAP y LIME.
 - **Reducir costos computacionales:**
 - Investigar formas de optimizar métodos como Boosting y Rotation Forest para hacerlos más accesibles en aplicaciones de tiempo real.
 - **Extender el alcance de los metaclasificadores:**
 - Adaptarlos a tareas de aprendizaje no supervisado, regresión o aprendizaje por refuerzo para ampliar sus aplicaciones más allá de la clasificación.

Sesión de preguntas

¡Gracias por su atención!



POLITÉCNICA

UNIVERSIDAD
POLÍTÉCNICA
DE MADRID



ESCUELA TÉCNICA
SUPERIOR DE INGENIEROS
INFORMÁTICOS